

WOJCIECH BRONIOWSKI
MATEMATYKA DYSKRETNĄ
WYKŁADY Z PRZYKŁADAMI W JĘZYKU PYTHON
Wyd. II

Wojciech Broniowski

Matematyka dyskretna, wyd. II

Wykłady z przykładami w języku Python

Sułkowice, wrzesień 2021

Na okładkach: Losowe grafy z zaznaczoną strukturą społeczną (Mathematica).

©by Wojciech Broniowski
Wydanie elektroniczne
ISBN 978-83-962099-1-7
e-mail: Wojciech.Broniowski@ujk.edu.pl
<https://www.ifj.edu.pl/~bronio>

Mojej Rodzinie

Spis treści

Przedmowa do wyd. II	IX
Przedmowa do wyd. I	XI
Wprowadzenie	XIII
0.1 Czym jest matematyka dyskretna?	XIII
0.2 Podstawowa literatura	XIV
1 Rekurencja	1
1.1 Wieże Hanoi	1
1.2 Notacja asymptotyczna	7
1.3 Liczby trójkątne	10
1.4 Liczby Fibonacciego	11
1.5 Równanie charakterystyczne	19
1.6 Rekurencja liniowa niejednorodna	23
1.7 Ruina gracza	27
1.8 Szybkie mnożenie i rekurencje „dziel i rządź”	34
Ćwiczenia	42
2 Kombinatoryka	47
2.1 Permutacje	48
2.2 Silnia	51
2.3 Wariacje	55
2.4 Permutacje z powtórzeniami	57
2.5 Kombinacje	58
2.6 Współczynniki dwumianowe	62
2.7 Współczynniki wielomianowe	65
2.8 Zasada szufladkowa	65
2.9 Układanie domina	67
Ćwiczenia	71
3 Elementy rachunku prawdopodobieństwa	73
3.1 Prawdopodobieństwo dyskretnie	74

3.2	Paradoks urodzin	79
3.3	Zasada włączania i wyłączania	84
3.4	Szaliki kibiców Korony Kielce	87
3.5	Problem sadowienia gości	91
3.6	Prawdopodobieństwo warunkowe	94
3.7	Twierdzenie Bayesa	95
	Ćwiczenia	98
4	Więcej kombinatoryki	101
4.1	Grupa permutacji	101
4.2	Liczby Stirlinga	103
4.3	Liczby Bella	107
4.4	Triangulacja wielokąta i liczby Catalana	110
4.5	Partycje liczb	113
4.6	Kompendium najczęstszych problemów kombinatorycznych .	121
	Ćwiczenia	122
5	Grafy	125
5.1	Spacer Eulera	127
5.2	Grafy Hamiltona	139
5.3	Kalejdoskop grafów	142
5.4	Izomorfizm grafów	144
5.5	Grafy planarne	146
5.6	Ille jest grafów?	152
5.7	Przeszukiwanie grafów	153
5.8	Grafy z wagami	155
5.9	Jeszcze o wieżach Hanoi	164
5.10	Drzewo Steinera	166
5.11	„Mały świat”	173
5.12	Kolorowanie wierzchołkowe grafów	176
5.13	Kolorowanie krawędziowe grafów	185
5.14	Liczby Ramseya	188
5.15	Kojarzenie małżeństw	190
5.16	Drzewa etykietowane	192
5.17	Notacja polska	197
5.18	Sieci zdarzeń	199
5.19	Przepływy w sieciach	202
	Ćwiczenia	208
6	Algorytmy	213
6.1	Czym jest algorytm	213
6.2	Automaty skończone	215
6.3	Maszyna Turinga	218
6.4	Problem 196	223

6.5	Lodziarz z Pińczowa i algorytm genetyczny	224
6.6	Klasyfikacja złożoności problemów decyzyjnych	230
	Ćwiczenia	238
7	Wskazówki i odpowiedzi do ćwiczeń	239
7.1	Rozdział 1	239
7.2	Rozdział 2	249
7.3	Rozdział 3	254
7.4	Rozdział 4	261
7.5	Rozdział 5	266
7.6	Rozdział 6	272
	Bibliografia	273
	Skorowidz	277
	Przykłady w języku Python	285

Przedmowa do wyd. II

Niniejsze wydanie zawiera uaktualnienia wyd. I [1] oraz bardzo istotną nową część, napisaną w formacie Jupyter Book, przedstawiającą programy w języku Python służące jako ilustracje do szeregu problemów omówionych w wykładzie. Mając na uwadze najnowsze tendencje literatury informatycznej, programy te stanowią tzw. *wykonywalną książkę* (executable book). Czytelnik może je urochomić w chmurze obliczeniowej „jednym pociśnięciem myszki” (bez kopiowania, instalacji, czy konfiguracji), modyfikować, zmieniać parametry, jednym słowem „uczyć się bawiąc”. Ta nowa jakość ma praktyczne znaczenie dydaktyczne: czytelnik poznaje teorię z pierwszej części konwencjonalnego „papierowego” podręcznika i od razu może przejść do implementacji programistycznej, dającej niebędne zrozumienie zagadnienia od strony informatycznej.

Podręcznik powstał dzięki wykładom i ćwiczeniom, prowadzonym przez autora dla studentów pierwszych lat informatyki oraz inżynierii danych na Uniwersytecie Jana Kochanowskiego w Kielcach w latach 2005–2020.

Autor pragnie podziękować dr Milenie Piotrowskiej z Uniwersytetu Jana Kochanowskiego w Kielcach za uwagi i korektę tekstu oraz Janowi Broniowskiemu za pomoc w utworzeniu wykonywalnej książki Jupyter Book.

Przedmowa do wyd. I

Niniejsza książka, przeznaczona dla studentów informatyki oraz wykładowców i prowadzących ćwiczenia, przedstawia najważniejsze i najpotrzebniejsze zagadnienia matematyki dyskretniej, koncentrując się na rekurencji, zaawansowanej kombinatoryce i klasycznym rachunku prawdopodobieństwa, grafach i algorytmach na grafach oraz sieciach, a także na wybranych elementach teorii algorytmów i ich złożoności. Ambicją autora jest ukazanie istoty tłumaczonego problemu w powiązaniu z elementarnym przeprowadzeniem poczatkującego studenta, po dzisiejszej szkole średniej zazwyczaj dość słabo obytego z technikami matematycznymi, poprzez szczegóły rachunkowe. Preferowaną metodą dydaktyczną jest, w miarę możliwości, ścieżka od szczegółu do ogółu: najpierw ukazywany i rozwiązywany jest pewien typowy problem z danego działu, a następnie przedstawiana jest ogólna teoria z definicjami, twierdzeniami, wnioskami, przykładami i uogólnieniami. W ten sposób wykładana abstrakcja znajduje wcześniejsze ugruntowanie w bardziej intuicyjnym przykładzie „z życia”, co powinno ułatwić zrozumienie i docenienie praktycznego znaczenia wykładanego materiału. Nienajlej, niektóre problemy, których wykład nie unika, łatwe nie są i należy przez nie mozolnie przebrnąć z kartką i ołówkiem w zaciszu domowym. Zgodnie z maksymą „uczyćć bawiąc” wiele zagadnień ma charakter rekreacyjny, co rekompensuje włożony trud w wymagających i może mniej atrakcyjnych, choć ważnych, częściach wykładu. Przy całym ukierunkowaniu na przystępność, wykład dotyczy matematyki i jej konkretnych zastosowań, zatem w jego śledzeniu niezbędna jest stosowna koncentracja i doza samozaparcia.

Wykład przeznaczony jest dla studentów pierwszego roku informatyki, wobec czego wymagana wiedza matematyczna niezbędna do jego śledzenia nie wykracza w zasadzie poza podstawowy program szkoły średniej. Warto jednak zawsze przypomnieć sobie ten materiał z podręczników szkolnych, w szczególności jego część *dyskretną*: podstawy logiki, teorii mnogości, kombinatoryki i rachunku prawdopodobieństwa. Materiał zahacza też nieco o zazwyczaj biegające równolegle lub z wyprzedzeniem standardowe wykłady algebry i analizy matematycznej, w szczególności używa elementarnej znajomości funkcji, układów równań liniowych, macierzy, ciągów i szeregów, obliczania granic, a także pewnych elementów rachunku różniczkowego i całkowego, jak rozwinięcie Taylora, całka gaussowska itp.

Zadaniem wykładu, oprócz przekazu stosownej porcji wiedzy, aparatu pojęciowego i typowych technik analizowania problemów matematyki dyskretniej, jest również nauczenie logicznego myślenia, „kombinowania”, sprytu matematycznego. Dlatego też na końcu każdego rozdziału podane są niekoniecznie łatwe problemy, mające na celu wciagnięcie studenta w tę pasjonującą matematykę. Wiele z tych zadań wymaga pomysłu, a ich rozwiązywanie częstokroć stawia na równi studenta i profesora, z tą jedyną różnicą, że profesor miał już okazję dłużej myśleć nad podobnymi zagadnieniami w swojej karierze! Wskazówki bądź rozwiązania trudniejszych zadań zawarte są na końcu książki.

I jeszcze jedno. Studenci, pamiętajcie:

Pytania nie są niedyskretne, odpowiedzi czasem tak.
Oscar Wilde

Wprowadzenie

0.1 Czym jest matematyka dyskretna?

Matematyka dyskretna stanowi swoisty wyciąg najpotrzebniejszych dla studen-
ta informatyki wiadomości z takich działów matematyki jak logika, teoria liczb,
arytmetyka modularna, teoria zbiorów przeliczalnych, topologia, teoria struk-
tur algebraicznych, teoria rekurencji, kombinatoryka, rachunek prawdopodobień-
stwa, teoria procesów stochastycznych, teoria grafów i sieci, optymalizacja proce-
sów, teoria algorytmów i ich złożoności czy wreszcie teoria informacji. Wspólną
cechą wszystkich tych dziedzin matematyki jest to, że zajmują się obiektyami
*dyskretnymi*¹, tj. oddzielonymi od siebie, tworzącymi struktury *nieciągłe*. Przed-
miot o tej właśnie nazwie zaczęto wykładać w latach 80. XX w. na uniwersyte-
tach amerykańskich jako „matematyczny niezbędnik” przyszłych informatyków,
dający im stosowne narzędzia i metody zawarte w jednym zwartym wykładzie
i dobrane pod kątem przydatności w ich dziedzinie. Od tego czasu *matematyka*
dyskretna jest standardem programowym na wszystkich studiach informatycz-
nych, chociaż wypełniająca ją treść może się znacznie różnić w zależności od
profilu uczelni, zaawansowania studentów, czasu trwania zajęć, podziału mate-
riału na poszczególne wykłady programu studiów czy wreszcie od indywidualnych
upodobań wykładowcy.

Cecha *dyskretności* oznacza, że w rozważanych zagadnieniach mamy do czynienia z układami przyjmującymi przeliczalne, czyli dające się ponumerować
liczbami naturalnymi, stany. Stany o wspólnych cechach można zliczać meto-
dami kombinatoryki, a w oparciu o wyniki obliczać prawdopodobieństwo zajścia
konkretnych, interesujących nas zdarzeń. Ponadto, układ może przechodzić od
stanu do stanu zgodnie z pewnymi regułami i taki proces możemy badać: jak
długo trwa, gdzie nas zaprowadzi i z jakim prawdopodobieństwem, czy możliwe
jest dotarcie do pewnego wyróżnionego stanu, jak taki proces zoptymalizować.
Niektóre z zastosowań matematyki dyskretnej poznamy w tym wykładzie, inne
można znaleźć w bardzo licznych specjalistycznych podręcznikach dotyczących

¹ (łac.) *discretus* – oddzielny

teorii algorytmów, baz danych, sieci komputerowych, automatyki, robotyki, telekomunikacji, kryptografii itd., gdzie omówiony tutaj podstawowy materiał będzie pomocny w zrozumieniu.

Matematyka dyskretna, obejmująca tak wiele podstawowych gałęzi Królowej Nauk, jest dziedziną otwartą, z wieloma słynnymi, prostymi do sformułowania, a dotąd nierożwiązanymi zagadnieniami. Za rozstrzygnięcie niektórych problemów czeka chwała i sława, tudzież ufundowano wysokie nagrody. Na przykład za udowodnienie, czy klasa tzw. algorytmów niedeterministycznych wielomianowych (NP) jest różna od klasy algorytmów wielomianowych (P), czy też jest z nią tożsama, co jest jednym z najważniejszych nierożwiązanych problemów matematyki dyskretnej o fundamentalnym znaczeniu, Instytut Matematyczny Claya² wypłaci milion dolarów! Nasz cel jest znacznie skromniejszy, a więc do dzieła...

0.2 Podstawowa literatura

Ogólnie dostępne polskojęzyczne podręczniki matematyki dyskretnej, zawierające szersze ujęcie wielu zagadnień tego wykładu, to:

- K. A. Ross, C. R. B. Write, *Matematyka dyskretna* [2],
- R. L. Graham, D. E. Knuth, O. Patashnik, *Matematyka konkretna* [3],
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Wprowadzenie do algorytmów* [4].

Pomocne mogą też być wykłady:

- A. Szepietowski, *Matematyka dyskretna* [5],

a w części dotyczącej teorii liczb i matematyki rekreacyjnej popularna książka

- J. H. Conway, R. K. Guy, *Księga liczb* [6].

Warto też sięgnąć po klasyczne podręczniki matematyczne, jak:

- W. Narkiewicz, *Teoria liczb* [7] oraz
- R. J. Wilson, *Wstęp do teorii grafów* [8].

Wiele przystępnych podręczników dotyczy zaawansowanej kombinatoryki [9–13], kopalnią wiedzy jest też monumentalny cykl

- D. E. Knuth, *Sztuka programowania* [14–18].

Standardowa literatura z elementami matematyki dyskretnej w języku angielskim to [19–28]. Po ukazaniu się I wydania niniejszej książki ukazały się też podręczniki po polsku [29–31].

² Instytut Matematyczny Claya – amerykańska organizacja utworzona w 1998 r., mająca na celu pogłębianie wiedzy matematycznej oraz jej popularyzację. W 2000 r. Instytut sformułował listę tzw. *siedmiu problemów milenijnych* (zob. <http://www.claymath.org/millennium>). Za rozwiązanie każdego z nich czeka nagroda w wysokości miliona dolarów.

Bardziej specjalistyczne pozycje literatury podawane są na bieżąco w kolejnych rozdziałach wykładu. Multum zadań jest przedstawionych i rozwiązań w [32].

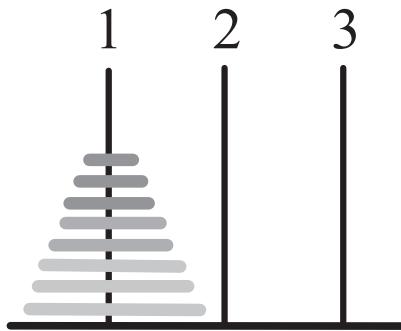
Rozdział 1

Rekurencja

W tym rozdziale pokażemy i rozwiążemy szczegółowo kilka bardzo znanych i reprezentatywnych problemów rekurencyjnych. Rekurencja stanowi trzon matematyki dyskretnej i programowania, bardzo wiele algorytmów ma bowiem składniki odwołujące się do samych siebie, czyli rekurencyjne. Zaczniemy od bardzo znanego problemu, opisanego w bodaj wszystkich podręcznikach matematyki dyskretnej, tj. problemu *Wież Hanoi*. Wprowadzimy użyteczną notację asymptotyczną. Następnie pobawimy się w układanie jednogroszówek i poznamy liczby trójkątne. Później ulegniemy magii liczb *Fibonacciego* i złotego podziału oraz przedstawimy ku przestrodze problem *ruiny gracza* w nadziei, że odstraszy od uprawiania hazardu! Podane zostaną ogólne metody rozwiązywania rekurencji liniowych, oparte na funkcjach tworzących oraz na równaniu charakterystycznym. Na koniec rozdziału przedstawimy rekurencje typu „dziel i rządź”, często występujące w zagadnieniach informatycznych, oraz odnoszące się do nich najważniejsze twierdzenia. Ilustracją będzie tu algorytm *szybkiego mnożenia* długich liczb.

1.1 Wieże Hanoi

Popularna łamigłówka, którą dawniej z łatwością można było kupić w sklepach z zabawkami, składa się z podstawki z przymocowanymi do niej trzema prętami



Rysunek 1.1: Wieże Hanoi

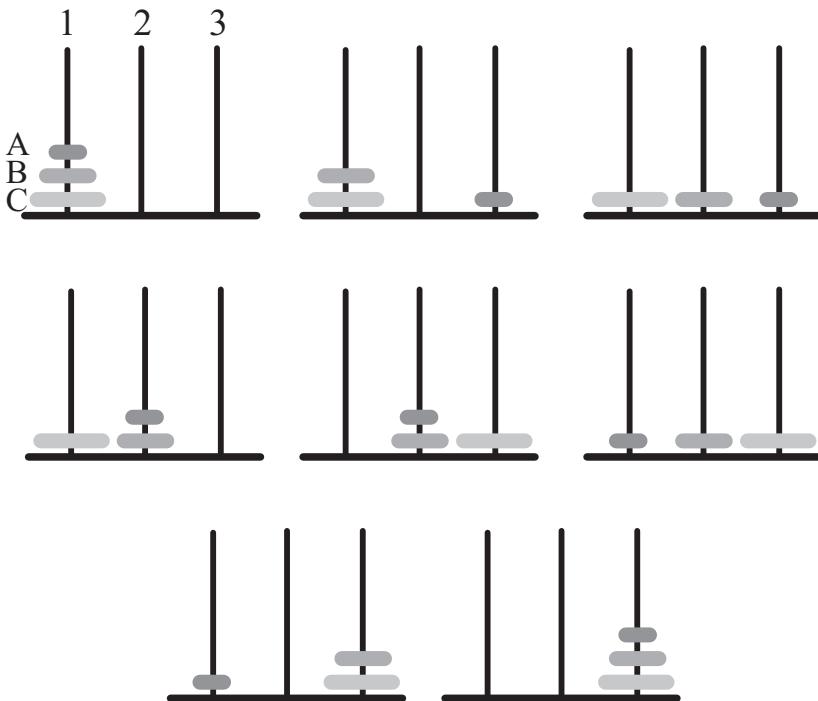
oraz kilku krążków o różnej średnicy z otworami, które można nanizać na pręty (rys. 1.1). Zabawka została wymyślona i spopularyzowana przez Édouarda Lucasa¹ w 1883 r. [33]. Tradycyjnie mamy $n = 8$ krążków, ale konkretna liczba nie jest tu istotna. Zadanie polega na tym, aby krążki początkowo nanizane na jeden pręt w kolejności od największego na spodzie do najmniejszego na górze przenieść na inny pręt. W danym ruchu możemy przenosić z pręta na pręt tylko po jednym krążku, ponadto nie wolno położyć większego krążka na mniejszym (patrz rys. 1.2). Oczywiście, zabronione jest też chowanie krążków pod łóżko czy do rękawa...

Powstają następujące zasadnicze pytania:

1. Czy da się to zrobić? Na przykład, gdyby pręty były tylko dwa, a nie trzy, to zadanie dla $n > 1$ jest niewykonalne. Pytamy więc o *istnienie rozwiązania* problemu.
2. Jeśli rozwiązanie istnieje, to jak je uzyskać, czyli jaki jest *algorytm* prowadzący do celu?
3. Jak długo trzeba wykonywać algorytm aby rozwiązać problem? Jaki jest najszybszy algorytm?
4. Jak szybko czas wykonywania algorytmu rośnie z liczbą krążków n (tzw. złożoność obliczeniowa)?
5. Jakie są uogólnienia problemu, jakie mają własności (patrz np. zad. 1.10).

W przypadku Wież Hanoi algorytm dający rozwiązanie jest bardzo prosty. Aby odgadnąć jego postać (dla dowolnego n), dobrze jest „pobawić się”, tj. rozwiązać problem dla małej wartości n i zauważać ogólną prawidłowość. Uproścmy więc sobie zadanie, zakładając na chwilę, że mamy tylko trzy krążki, $n = 3$. Oznaczmy pręty jako 1, 2 i 3 oraz krążki, od najmniejszego do największego, jako A , B i C . Początkowo na pręcie 1 mamy układ, wyliczając od dołu, CBA , natomiast pręty 2 i 3 są puste (patrz lewy górny rys. 1.2). Nasze zadanie to przeniesienie krążków

¹ François Édouard Anatole Lucas (1842-1891), matematyk francuski znany m.in. z badań nad liczbami Fibonacciego.



Rysunek 1.2: Kolejne kroki algorytmu Wież Hanoi dla trzech krążków ($n = 3$). Po siedmiu ruchach problem zostaje rozwiązyany: krążki są przeniesione z pręta A na pręt C

z pręta 1 na pręt 3. Kolejne kroki pokazane są na rys. 1.2. Najpierw przenosimy krążek A na pręt 3. W następnym ruchu przenosimy B na 2, potem A na 2, następnie C na 3. Mamy już podstawę naszej nowej budowli! Teraz przenosimy A na 1, B na 3 i wreszcie A na 3. A więc poszło dość łatwo, siedem ruchów² i fajrant. Dużo bardziej zwięzły zapis kolejnych kroków algorytmu przedstawiony jest w tabeli 1.1.

Zastanówmy się teraz nad jeszcze łatwiejszym przypadkiem, a mianowicie $n = 2$. Patrzmy na cztery pierwsze wiersze tabeli 1.1, do trzeciego ruchu włącznie, ponadto zignorujmy krążek C . Widzimy, że w takim przypadku tabela opisuje zredukowany problem przeniesienia dwóch krążków A i B z pręta 1 na pręt 2, na co wystarczają trzy ruchy. Chociaż ten fakt brzmi banalnie, jest w nim bardzo istotne prawidło: proces przeniesienia trzech krążków zawiera w sobie podproces przeniesienia dwóch krążków. Patrzmy teraz na dolne cztery wiersze tabeli, ponownie ignorując krążek C . Znowu mamy tu przedstawiony podproces przeniesienia krążków A i B , tym razem z pręta 2 na pręt 3, ponownie w trzech ruchach. Robimy więc kluczową obserwację: proces przeniesienia trzech krążków z 1 na 3 składa się z podprocesu przeniesienia dwóch górnego krążków z 1 na 2 (ruchy 1, 2, 3), przeniesienia dolnego krążka z 1 na 3 (ruch 4) oraz z podprocesu przeniesienia dwóch krążków z 2 na 3 (ruchy 5, 6, 7). Głębia tej obserwacji tkwi

² Rysunków jest 8, ale przejść między nimi 7.

Tabela 1.1: Rozwiązywanie problemu Wież Hanoi dla trzech krążków. Kolumny pokazują stan prętów 1, 2 i 3 po kolejnych ruchach. Zapis CBA oznacza, że krążek C leży na dole, B na nim, a A na górze itd.

Ruch	1	2	3
0	CBA		
1	CB		A
2	C	B	A
3	C	BA	
4		BA	C
5	A	B	C
6	A		CB
7			CBA

w tym, że taki rozkład na podprocesy zachodzi dla dowolnej liczby krążków n , co zaraz udowodnimy za pomocą znanej ze szkoły średniej zasady indukcji matematycznej.

Oznaczmy przez A_n wykonanie algorytmu dla n krążków³.

Tw. 1.1. *Problem Wież Hanoi ma rozwiązanie.*

Dowód, oparty na indukcji matematycznej, polega na wskazaniu algorytmu dającego rozwiązanie problemu:

Alg. 1.1 (algorytm Hanoi). *Dla $n = 1$ istnienie rozwiązania jest oczywiste. Założymy, że istnieje rozwiązanie dla n . Algorytm A_{n+1} jest następujący: wykonaj A_n , przenosząc n górnych krążków z pręta 1 na 2, następnie przenieś największy krążek na pręt 3, na koniec wykonaj A_n przenosząc krążki z pręta 2 na 3. Przez wskazanie A_{n+1} istnieje. Na mocy zasady indukcji matematycznej A_n istnieje dla każdego $n \geq 1$.*

Teraz zastanówmy się, jaka jest najmniejsza liczba ruchów potrzebna na rozwiązanie problemu, którą oznaczymy jako h_n . Z konstrukcyjnego dowodu⁴ Tw. 1.1 widzimy natychmiast, że dla podanego algorytmu

$$\begin{aligned} h_1 &= 1, \\ h_{n+1} &= h_n + 1 + h_n = 2h_n + 1, \quad n \geq 1. \end{aligned} \tag{1.1}$$

³ Wskaźnik (indeks) dolny n numeruje kolejne procesy A i jest użyty podobnie do znanego ze szkoły numerowania kolejnych wyrazów ciągu liczbowego. W obecnym przypadku A_1 oznacza algorytm przeniesienia jednego krążka, A_2 dwóch krążków, A_3 trzech (jak na rys. 1.2) itd.

⁴ Dowód konstrukcyjny to taki dowód, który podaje konstrukcję konkretnego rozwiązania problemu. Z praktycznego punktu widzenia dowody takie są bardzo cenne. Dowody niekonstrukcyjne poprzestają na dowiedzeniu istnienia bądź braku istnienia rozwiązania, ale nie podają, jak w przypadku istnienia je znaleźć.

Pierwsze h_n bierze się z pierwszego wykonania A_n (przeniesienie n krążków z preta 1 na preć 2, największy krążek pozostaje w spoczynku), jedynka z przeniesienia największego krążka na preć 3, a drugie h_n z drugiego wykonania A_n (przeniesienie n krążków na największy krążek na precie 3). Mamy więc ważny wzór, wyrażający liczbę kroków algorytmu dla $n + 1$ krążków z pomocą liczby kroków dla n krążków.

Można zadać pytanie, czy przypadkiem nie istnieje algorytm o mniejszej liczbie ruchów. Konstrukcja w dowodzie Tw. 1.1 podała jedno rozwiązanie, ale skąd wiemy, czy nie istnieje rozwiązanie szybsze? A więc ktoś mógłby powiedzieć, że tak naprawdę wiemy teraz tylko tyle, że najmniejsza liczba kroków spełnia nierówność $h_{n+1} \leq 2h_n + 1$, czyli mamy górne ograniczenie na h_{n+1} . Chwila zastanowienia pokazuje jednak, że jest to również ograniczenie dolne. W pewnym momencie wykonywania (dowolnego) algorytmu musimy przenieść krążek o numerze $n + 1$ (największy). Aby móc to zrobić, musielibyśmy wcześniej przenieść n krążków, na co zużylibyśmy co najmniej h_n ruchów, a po przeniesieniu krążka $n + 1$ musimy nań nałożyć z powrotem n krążków, co znowu wymaga co najmniej h_n ruchów (co najmniej, bo zawsze możemy bawić się, przenosząc jakiś krążek wte i wewte, tracąc czas). Tak więc zachodzi również nierówność $h_{n+1} \geq 2h_n + 1$ i ostatecznie algorytm o najmniejszej możliwej liczbie ruchów spełnia równanie (1.1). Całkiem inny dowód faktu, że algorytm 1.1 jest w istocie najszybszy z możliwych podamy w oparciu o silne techniki teorii grafów w rozdz. 5.9.

W wyniku naszej analizy zamieniliśmy naszą zabawkę na problem czysto matematyczny, wyrażony równaniem (1.1). Jest to przykład *rekurencji*, czyli równania, w którym po prawej stronie pojawia się ta sama wielkość co po lewej stronie, tyle, że dla mniejszych wartości wskaźnika n . A oto formalna definicja:

Def. 1.1. *Ciągiem rekurencyjnym nazywamy ciąg podany przepisem*

$$a_{n+1} = F(a_n, a_{n-1}, \dots, a_{n-k+1}; n),$$

gdzie funkcja F zależy od wyrazów a_n, \dots, a_{n-k+1} oraz w ogólności może zależeć od n . Liczbę k nazywamy rzędem rekurencji. Ponadto, aby określić rekurencję, zadajemy k pewnych wyrazów ciągu.

Dla rekurencji Hanoi funkcja F ma postać $F(h_n; n) = 2h_n + 1$, zatem $k = 1$ – jest to rekurencja rzędu pierwszego. Musimy zadać jeden wyraz, w naszym przypadku $h_1 = 1$, aby jednoznacznie określić ciąg h_n .

Teraz będziemy chcieli *rozwiązać* rekurencję, czyli podać wzór na h_n zawierający jedynie n . Oczywiście, aby znaleźć wyrazy ciągu h_n , zawsze możemy postąpić rekurencyjnie, czyli nawet dla bardzo dużych n , powiedzmy $n = 1000$, powyliczać kolejno wszystkie h_n , począwszy od $n = 2, 3, 4, \dots$ aż do 1000. Jest to w zasadzie równoważne wykonaniu algorytmu i policzeniu liczby wykonanych kroków, z czym jest dużo pracy. Nie o to nam jednak chodzi. Pragniemy bowiem znaleźć tzw. *postać ogólną* dla h_n , czyli wzór zawierający tylko n .

Def. 1.2. Rozwiązań rekurencji zadanej danym ciągiem rekurencyjnym to po danej jego postaci ogólnej.

Inaczej mówiąc, znając n , „od razu” będziemy wiedzieć, ile pracy nas czeka przy przenoszeniu kramków (ile wynosi h_n).

W jaki sposób rozwiązać rekurencję Wież Hanoi? Jest kilka sposobów. Najprostszy to odgadnięcie wyniku, a następnie jego dowiedzenie – sama zgadywanka, rzecz jasna, matematykowi nie wystarczy! Jest to zatem metoda „zgadnij i sprawdź”. Zgadywanie zawsze rozpoczynamy od wypisania kilku początkowych wyrazów ciągu h_n , które obliczamy ze wzoru (1.1) dla kolejnych wartości n :

$$\begin{array}{ccccccc} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & \dots \\ 1 & 3 & 7 & 15 & 31 & 63 & \dots \end{array}$$

Następnie szukamy prawidłowości w otrzymanym ciągu. Przydatna jest tu „przyjaźń z liczbami” – musimy je znać, lubić i podświadomie rozpoznawać! W naszym przypadku widzimy, że gdyby liczby były o jeden większe, mielibyśmy do czynienia z kolejnymi potęgami dwójki: $2, 4, 8, 16 \dots$, zatem po prostu

$$h_n = 2^n - 1. \quad (1.2)$$

Jeśli tego jeszcze nie widzimy, wypisujemy więcej wyrazów, jeśli ciągle nic nam nie swita, to kupujemy w sklepie z zabawkami Wieże Hanoi ... lub czytamy ten rozdział dalej. Zrobiliśmy „zgadnij”, pozostało „sprawdzić” rekurencję (1.1).

Tw. 1.2. Równanie (1.2) jest rozwiązaniem rekurencji (1.1).

Dowód: Rzeczywiście, $h_1 = 2^1 - 1 = 1$ oraz $h_{n+1} = 2^{n+1} - 1 = 2(2^n - 1) + 1 = 2h_n + 1$, co kończy sprawdzanie. \square

Nie zawsze łatwo jest odgadnąć rozwiązanie. Inny przydatny sposób rozwiązywania rekurencji to metoda „podstaw i rozwini”. Składa się ona z kilku kroków. Najpierw podstawiamy do prawej strony wzoru (1.1) tę samą rekurencję, ale dla n o jeden mniejszego, następnie czynność tę powtarzamy kilkukrotnie. Możemy ten krok nazwać *iterowaniem rekurencji*. Wynik kolejnych podstawień jest następujący:

$$\begin{aligned} h_{n+1} &= 1 + 2h_n = \\ &= 1 + 2(1 + 2h_{n-1}) = 1 + 2 + 4h_{n-1} = \\ &= 1 + 2 + 4(1 + 2h_{n-2}) = 1 + 2 + 4 + 8h_{n-2} = \\ &= 1 + 2 + 4 + 8(1 + 2h_{n-3}) = 1 + 2 + 4 + 8 + 16h_{n-3}. \end{aligned}$$

Teraz zauważamy ogólny wzorzec:

$$h_{n+1} = 1 + 2 + 4 + \dots + 2^i + 2^{i+1}h_{n-i},$$

gdzie $i = 1, 2, \dots, n - 1$. Weźmy $i = n - 1$, aby wskaźnik ostatniego wyrazu wyniósł $n - i = 1$. Wtedy

$$h_{n+1} = 1 + 2 + 4 + \cdots + 2^{n-1} + 2^n h_1 = 1 + 2 + 4 + \cdots + 2^{n-1} + 2^n = \sum_{i=0}^n 2^i.$$

Ponieważ, jak pamiętamy ze szkoły, suma częściowa ciągu geometrycznego wynosi

$$\sum_{i=0}^n a^i = \frac{a^{n+1} - 1}{a - 1}, \quad (1.3)$$

zatem kładąc $a = 2$, otrzymujemy ponownie znany już wynik (1.2). Ogólna teoria ciągów rekurencyjnych, którą poznamy w dalszej części tego rozdziału, dostarcza jeszcze innych metod rozwiązyania problemów rekurencyjnych.

Posiadłszy wzór (1.2), możemy natychmiast otrzymać liczbę kroków potrzebną na rozwiązanie problemu dla dowolnego n . Na tym m.in. polega przewaga rozwiązania ogólnego nad rekurencyjnym. Legenda o Wieży Brahmy, będąca wersją wieży Hanoi o $n = 64$ (pomijamy matematycznie nieistotny fakt, że krążki są ze złota, a pręty z diamentów), głosi, że gdy mnisi buddyjscy przenoszą wszystkie krążki, to nastąpi koniec świata. Policzymy, ile im to zajmie. Z naszego rozwiązania ogólnego dostajemy

$$h_{64} = 2^{64} - 1 = 18446744073709551615.$$

Założymy, że mnisi mają już wielką wprawę i przeniesienie jednego krążka z pręta na pręt zajmuje im tylko 5 sekund. Oczywiście pracują bez wytchnienia. Wtedy wykonanie całego algorytmu potrwa ok. $3 \cdot 10^{12}$ lat! Zważywszy, że dotychczasowy wiek Wszechświata to „tylko” $(13.7 \pm 0.2) \cdot 10^9$ lat, jesteśmy bezpieczni – mnisi będą pracować 200 razy dłużej! Pracując w tempie mnichów, można rozwiązać problem Wież Hanoi ($n = 8$) w ok. 20 minut. Widzimy więc, że czas wykonywania algorytmu Hanoi wzrasta bardzo szybko z n . Jest to wzrost wykładniczy (eksponencjalny), ponieważ $h_n = 2^n - 1 \sim 2^n$. Znak \sim oznacza tu, że dla dużych n obie strony są niemal równe, tj. ich iloraz dąży do stałej różnej od zera przy $n \rightarrow \infty$ (patrz następny podrozdział). Klasa algorytmów tego typu oznaczana jest jako EXP i są to algorytmy trudne do wykonania (powolne) w tym sensie, że wzrost długości przetwarzanych danych (w tym przypadku liczby krążków n) przekłada się na bardzo szybki (wykładniczy) wzrost czasu wykonywania algorytmu (klasyfikację złożoności algorytmów omówimy w rozdz. 6.6).

1.2 Notacja asymptotyczna

Spotkaliśmy się właśnie po raz pierwszy z porównywaniem zachowania ciągów w granicy asymptotycznej, czyli przy $n \rightarrow \infty$. Będzie nam to wielokrotnie potrzebne w tym wykładzie. Bardziej ogólnie, możemy porównywać funkcje zmiennej rzeczywistej x przy $x \rightarrow \infty$. Zawęzimy się do przypadku funkcji o dodatnich

wartościach,

$$f(x) > 0, \quad g(x) > 0.$$

Def. 1.3. Symbol $f(x) \sim g(x)$ oznacza, że

$$\exists c > 0 : \lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = c.$$

Innymi słowy, obliczając iloraz funkcji $f(x)/g(x)$ dla coraz większych wartości argumentu, dostaniemy w granicy dodatnią stałą.

Uwaga 1.1. Niektórzy autorzy definiują $f(x) \sim g(x)$ ze stałą $c = 1$, tj.

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 1.$$

Ponieważ ciąg jest zawężeniem funkcji zmiennej rzeczywistej do dziedziny liczb naturalnych, powyższa definicja przenosi się na ciągi o wyrazach dodatnich, gdzie piszemy np. $a_n \sim b_n$, co oznacza $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = c$, ($c > 0$). Zauważmy, że definicja stosuje się do funkcji czy ciągów, które mogą być rozbieżne do nieskończoności, np. piszemy $n \sim n + 1$, bo $\lim_{n \rightarrow \infty} n/(n + 1) = 1$, natomiast $\lim_{n \rightarrow \infty} n = \infty$ oraz $\lim_{n \rightarrow \infty} (n + 1) = \infty$.

Często używanym symbolem jest \mathcal{O} (O-duże):

Def. 1.4. Mówimy, że $f(x) = \mathcal{O}(g(x))$ („ $f(x)$ jest O-duże od $g(x)$ ”) $\iff \exists C > 0 \exists x_0 \forall x > x_0 : Cg(x) \geq f(x)$.

Funkcja $g(x)$ przemnożona przez pewną dodatnią stałą jest więc *górnym ograniczeniem* dla $f(x)$ dla wszystkich *dostatecznie dużych* (asymptotycznych) wartości x . Analogicznie definiujemy ograniczenie dolne:

Def. 1.5. $f(x) = \Omega(g(x)) \iff \exists c > 0 \exists x_0 \forall x > x_0 : f(x) \geq cg(x)$,

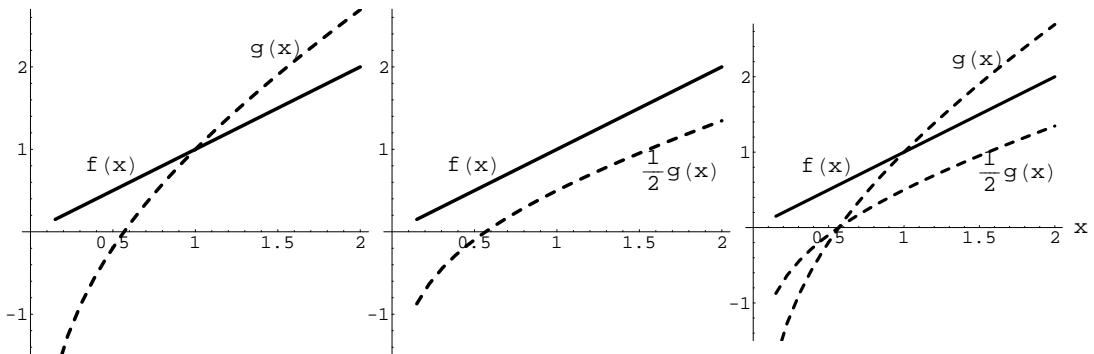
oraz ograniczenie obustronne:

Def. 1.6. $f(x) = \Theta(g(x)) \iff \exists C > 0, c > 0 \exists x_0 \forall x > x_0 : Cg(x) \geq f(x) \geq cg(x)$.

Wniosek 1.1. Jeśli $f(x) = \mathcal{O}(g(x))$ i $f(x) = \Omega(g(x))$, to $f(x) = \Theta(g(x))$.

Sytuacja z tych trzech definicji zilustrowana jest na rys. 1.3 dla przykładowych funkcji⁵ $f(x) = x$ oraz $g(x) = x + \log x$. Pokażemy najpierw, że $f(x) = \mathcal{O}(g(x))$

⁵ Symbol $\log x$ oznacza w tej książce logarytm naturalny, tj. $\log x = \log_e x$, często oznaczany w literaturze jako $\ln x$.



Rysunek 1.3: Ilustracja symboli \mathcal{O} , Ω i Θ : $f(x) = \mathcal{O}(g(x))$ (lewa strona), $f(x) = \Omega(g(x))$ (środek) oraz $f(x) = \Theta(g(x))$ (prawa strona)

(lewa część rysunku). Przy wyborze stałych C i x_0 mamy dowolność. Weźmy $C = 1$ oraz $x_0 = 1.5$. Istotnie, dla $x > x_0$ mamy $x + \log x > x$, bo $\log x > 0$ dla $x > 1$. Dla pokazania $f(x) = \Omega(g(x))$ bierzemy $c = 1/2$ i $x_0 = 1.5$. Musimy pokazać, że dla $x > x_0$ zachodzi $x \geq 1/2(x + \log x)$, co jest równoważne $\log x < x$, lub $x < e^x$, co jest spełnione dla wszystkich liczb rzeczywistych (dodatnich).

Wprost z definicji wynikają następujące fakty: $f(x) = \mathcal{O}(g(x)) \Leftrightarrow g(x) = \Omega(f(x))$ oraz $f(x) = \Theta(g(x)) \Leftrightarrow g(x) = \Theta(f(x))$. Mamy też przechodniość: $f(x) = \Theta(g(x)) \wedge g(x) = \Theta(h(x)) \Rightarrow f(x) = \Theta(h(x))$, zatem symbol $\Theta(g(x))$ definiuje klasę równoważności funkcji, które asymptotycznie zachowują się „tak samo”. Zauważmy, że symbol \sim nie jest równoważny symbolowi Θ . Weźmy bowiem $f(x) = 2 + \sin x$ i $g(x) = 1$. Funkcja $f(x)$ oscyluje między wartościami 1 a 3. Biorąc $C = 4$, $c = 1/2$ i $x_0 = 0$ sprawdzamy natychmiast, że $f(x) = \mathcal{O}(g(x))$. Nie jest natomiast prawda, że $f(x) \sim g(x)$, bo np. granica $\lim_{x \rightarrow \infty} f(x)/g(x) = \lim_{x \rightarrow \infty}(2 + \sin x)$ nie istnieje.

Podajmy jeszcze do kompletu symbol o (o małe):

Def. 1.7. Mówimy, że $f(x) = o(g(x))$ („ $f(x)$ jest o-małe od $g(x)$ ”) w punkcie x_0 , jeśli

$$\lim_{x \rightarrow x_0} \frac{f(x)}{g(x)} = 0.$$

W szczególności w badaniu zachowania asymptotycznego za x_0 bierzemy ∞ .

Na koniec pewna istotna przestroga odnośnie do wprowadzonej notacji. Znak $=$ we wszystkich powyższych definicjach nie oznacza równości, tylko przynależność do klasy funkcji. Na przykład $f(x) = \mathcal{O}(g(x))$ oznacza, że $f(x)$ należy do zbioru tych wszystkich funkcji, które są asymptotycznie ograniczone od góry przez $Cg(x)$. Niemniej znak $=$ jest wygodny, gdyż pozwala na zapis np. $f(x) = x^2 + \mathcal{O}(x)$, co tak naprawdę oznacza $f(x) - x^2 = \mathcal{O}(x)$, czyli że różnica $f(x) - x^2$ jest asymptotycznie ograniczona przez Cx .

1.3 Liczby trójkątne

Powróćmy teraz do poznawania rekurencji i rozważmy inny problem tego typu.

Def. 1.8 (liczby trójkątne). *Bawimy się jednogroszówkami i układamy z nich coraz większe „trójkąty”, jak na rys. 1.4. Oznaczmy liczbę monet użytych do zbudowania podstawy jako n . Jaka jest całkowita liczba p_n monet w trójkącie?*

Zauważamy, że trójkąt o numerze n powstaje z trójkąta o numerze $n - 1$ poprzez dodanie podstawy z n jednogroszówek. W ten sposób otrzymujemy prostą rekurencję

$$\begin{aligned} p_1 &= 1, \\ p_n &= p_{n-1} + n, \end{aligned} \tag{1.4}$$

z początkowymi wyrazami

$$\begin{array}{ccccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & \dots \\ 1 & 3 & 6 & 10 & 15 & 21 & \dots \end{array}$$

W notacji z definicji 1.1 mamy teraz

$$F(p_{n-1}; n) = p_{n-1} + n. \tag{1.5}$$

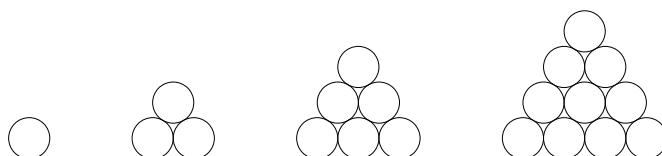
Rekurencja jest więc rzędu 1 i zależy w jawnym sposób od n .

Odgadnięcie rozwiązania (czyli wyrazu ogólnego na p_n) nie jest na pierwszy rzut oka oczywiste, więc w celu rozwiązania zastosujmy wprowadzoną w poprzednim podrozdziale metodę „podstaw i rozwiń”. Dostajemy

$$\begin{aligned} p_n &= n + p_{n-1} = n + (n - 1) + p_{n-2} \\ &= n + (n - 1) + (n - 2) + p_{n-3} = \\ &= n + (n - 1) + (n - 2) + (n - 3) + p_{n-4} = \\ &\dots \\ &= n + (n - 1) + (n - 2) + (n - 3) + \dots + (n - i) + p_{n-i-1}. \end{aligned}$$

Przy doborze i kierujemy się zasadą, aby ostatni wypisany wyraz, p_{n-i-1} , miał wskaźnik 1, ponieważ wyraz początkowy $p_1 = 1$ znamy. Daje to równanie $n - i - 1 = 1$, skąd $i = n - 2$. Wtedy

$$p_n = n + (n - 1) + \dots + 3 + 2 + p_1 = n + (n - 1) + \dots + 3 + 2 + 1.$$



Rysunek 1.4: Zabawa w układanie trójkątów z jednogroszówek

Stosując znany ze szkoły wzór na sumę częściową ciągu arytmetycznego,

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}, \quad (1.6)$$

otrzymujemy końcowy wynik

$$p_n = \frac{n(n+1)}{2}. \quad (1.7)$$

Wyobraźmy sobie, że dajemy komuś trójkąt z grosików o podstawie długości n i każemy mu je policzyć „na piechotę”. Czas wykonywania tego „algorytmu” jest *wielomianowy*, co określamy mianem „szybki”: liczba otrzymanych kawałków rośnie dla dużych n jak kwadrat długości podstawy n (dzielony przez 2). Różnica zachowania p_n i ciągu Hanoi $h_n \sim 2^n$ wynika z różnej postaci funkcji F w definicji 1.1. W podrozdziale 1.5 zrozumiemy dokładniej, dlaczego tak jest.

1.4 Liczby Fibonacciego

Bez wątpienia najsłynniejszym ciągiem rekurencyjnym, rozważanym osiem stuleci temu jako rozwiązywanie modelu rozmnażania się królików, jest ciąg Fibonacciego,⁶ [3, 33]. Ciąg ten był też parę wieków wcześniej znany matematykom hinduskim [34]. Zdefiniowany jest w taki sposób, że jego dwa pierwsze wyrazy są równe jedności, a każdy następny jest sumą dwóch poprzednich:

$$\begin{aligned} F_1 &= 1, F_2 = 1, \\ F_n &= F_{n-1} + F_{n-2}, \quad n > 2. \end{aligned} \quad (1.8)$$

Rekurencja sięga więc dwa wyrazy wstecz – jest rzędu drugiego. Wypiszmy kilkanaście początkowych wyrazów:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584 \dots \quad (1.9)$$

Otrzymujemy zatem silnie rosnący ciąg liczb naturalnych. Pierwsze wyzwanie to *rozwiązańe rekurencji*. W odróżnieniu od uprzednio rozważanych problemów, odgadnięcie rozwiązania nie jest łatwe, również metoda „podstaw i rozwiń” nie prowadzi do celu.

Bardzo eleganckim i silnym sposobem, który teraz poznamy, jest użycie tzw. *funkcji tworzącej*.

Def. 1.9. *Rozważmy ciąg liczbowy (a_n) , $n \geq 0$. Funkcja tworząca tego ciągu to szereg postaci*

$$f(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + \dots = \sum_{n=0}^{\infty} a_n z^n,$$

⁶ Fibonacci, Leonardo z Pizy (ok. 1170-1250), włoski matematyk, odkrywca liczb Fibonacciego, spopularyzował system pozycyjny (arabski) w Europie.

gdzie w ogólności z jest liczbą zespoloną⁷. Zauważmy, że funkcja tworząca niesie dokładnie tę samą informację, co ciąg (a_n) . Istotnie, dla danego ciągu tworzymy funkcję $f(z)$ zgodnie z definicją 1.9. Natomiast odwrotnie, mając funkcję tworzącą $f(z)$ otrzymujemy kolejne wyrazy ciągu poprzez n -krotne różniczkowanie w $z = 0$:

$$a_0 = f(0), \quad a_1 = f'(0), \quad a_2 = \frac{1}{2!}f''(0), \quad a_3 = \frac{1}{3!}f'''(0), \quad \dots, \quad a_n = \frac{1}{n!}f^{(n)}(0)$$

(n -ty wyraz to n -ta pochodna w $z = 0$ podzielona przez $n!$). Dowód powyższego równania opiera się na n -krotnym obliczeniu pochodnej $f(z)$, a następnie położeniu $z = 0$. Korzystamy tu z faktu, że $(z^k)' = kz^{k-1}$, zatem dla n -tej pochodnej

$$(z^n)^{(n)} = n(n-1)(n-2)\dots 2 \cdot 1 = n!$$

oraz

$$(z^n)^{(k)} = n(n-1)(n-2)\dots(n-k+1)z^{(n-k)}$$

dla $k < n$.

Jako bardzo prosty przykład wprowadzonej definicji rozważmy ciąg stały o wyrazach $c_n = 1$. Wówczas funkcja tworząca dana jest przez szereg geometryczny

$$f(z) = \sum_{n=0}^{\infty} c_n z^n = 1 + z + z^2 + z^3 + \dots = \frac{1}{1-z}.$$

Prawa strona postaci $1/(1-z)$ „koduje” w tym przypadku informację o ciągu stałym. Łatwo sprawdzić, że

$$\left(\frac{1}{1-z}\right)^{(n)} = n! \frac{1}{(1-z)^n},$$

zatem dla $z = 0$ mamy $f^{(n)}(0) = n!$ i istotnie widzimy, że

$$c_n = \frac{f^{(n)}(0)}{n!} = 1.$$

Powróćmy teraz do naszego wyjściowego problemu i utwórzmy formalnie funkcję tworzącą dla liczb Fibonacciego F_n :

$$f(z) = F_1 z + F_2 z^2 + \dots = \sum_{n=1}^{\infty} F_n z^n. \quad (1.10)$$

⁷ Na mocy znanego z analizy matematycznej Tw. Cauchy'ego-Hadamarda o zbieżności szeregow potęgowych (zob. [35] lub inny podręcznik analizy matematycznej), konstrukcja z def. (1.9) ma sens dla

$$|z| < \frac{1}{\limsup_{n \rightarrow \infty} \sqrt[n]{|a_n|}}.$$

Jednak w zagadnieniach kombinatorycznych funkcję tworzącą można traktować formalnie, tj. bez zwracania uwagi na zbieżność [3].

Zauważmy, że konwencjonalnie sumujemy tu od $n = 1$, a nie od $n = 0$, jak w definicji 1.9, co oczywiście możemy zrobić (możemy też przyjąć $F(0) = 0$ i sumować od $n = 0$). Bardzo częstą sztuczką jest przesunięcie indeksu sumowania zgodnie z regułą

$$\sum_{n=k}^p c_n = \sum_{n=k+m}^{p+m} c_{n-m}, \quad (1.11)$$

gdzie m jest dowolną liczbą całkowitą. Prawdziwość tego wzoru jest oczywista, jeśli wypiszemy jawnie lewą i prawą stronę, które są sobie równe i wynoszą $c_k + c_{k+1} + \dots + c_{p-1} + c_p$.

Napiszmy teraz wzory wynikające z (1.10), gdzie po prawej stronie przesuwamy indeks sumowania:

$$\begin{aligned} zf(z) &= F_1 z^2 + F_2 z^3 + \dots = \sum_{n=1}^{\infty} F_n z^{n+1} = \sum_{n=2}^{\infty} F_{n-1} z^n, \\ z^2 f(z) &= F_1 z^3 + F_2 z^4 + \dots = \sum_{n=1}^{\infty} F_n z^{n+2} = \sum_{n=3}^{\infty} F_{n-2} z^n. \end{aligned} \quad (1.12)$$

Metoda otrzymania równania na funkcję tworzącą jest następująca: mnożymy obie strony równania (1.8) przez z^n , po czym sumujemy po n , począwszy od $n = 3$ do nieskończoności. Wartość dolnej granicy sumowania wynika z obecności F_{n-2} po prawej stronie (1.8). Sumując od $n = 3$, dostajemy członki z F_1, F_2 itd., natomiast gdybyśmy sumowali od mniejszych n , np. $n = 2$, otrzymalibyśmy człon z F_0 , a ten wyraz ciągu nie jest określony i nie wiedzielibyśmy co zrobić. Podana procedura jest więc następująca:

$$\begin{aligned} F_n &= F_{n-1} + F_{n-2}, \\ F_n z^n &= F_{n-1} z^n + F_{n-2} z^n, \\ \sum_{n=3}^{\infty} F_n z^n &= \sum_{n=3}^{\infty} F_{n-1} z^n + \sum_{n=3}^{\infty} F_{n-2} z^n, \\ \sum_{n=1}^{\infty} F_n z^n - F_1 z - F_2 z^2 &= \sum_{n=2}^{\infty} F_{n-1} z^n - F_1 z^2 + \sum_{n=3}^{\infty} F_{n-2} z^n, \end{aligned} \quad (1.13)$$

gdzie w ostatnim wierszu poszerzyliśmy dolne granice sumowania, co zostało skompensowane odjęciem odpowiednich składników, np.

$$\sum_{n=3}^{\infty} F_n z^n = \sum_{n=1}^{\infty} F_n z^n - F_1 z - F_2 z^2.$$

Następnie używamy wzorów (1.10, 1.12) i otrzymujemy z (1.13)

$$f(z) - F_1 z - F_2 z^2 = zf(z) + z^2 f(z) - F_1 z^2,$$

czyli

$$f(z) - zf(z) - z^2 f(z) = F_1 z + (F_2 - F_1)z^2.$$

Wstawiając warunki początkowe $F_2 = F_1 = 1$, dostajemy w wyniku bardzo proste równanie

$$f(z) - zf(z) - z^2 f(z) = z,$$

zatem ostatecznie

$$f(z) = \frac{z}{1 - z - z^2}. \quad (1.14)$$

W tym momencie znamy funkcję tworzącą naszego problemu, mamy więc wszelką информацию, aby znaleźć liczby F_n . Zamiast podanego wcześniej przepisu z różniczkowaniem, który jest kłopotliwy dla dalszych wyrazów, zastosujemy inną metodę pozwalającą „wyłuskać” wzór ogólny dla F_n . W tym celu wykonamy krok techniczny, mianowicie rozkład wyrażenia (1.14) na *ułamki proste*, z czym jest nieco algebraicznej pracy. Chcemy zapisać funkcję tworzącą jako

$$f(z) = \frac{A}{1 - uz} + \frac{B}{1 - vz}, \quad (1.15)$$

ponieważ wyrażenie tej postaci umiemy rozłożyć w szereg geometryczny. Porównując wzór (1.14) i sprowadzone do wspólnego mianownika wyrażenie (1.15), otrzymujemy

$$\frac{A + B - (Av + Bu)z}{(1 - uz)(1 - vz)} = \frac{z}{1 - z - z^2}, \quad (1.16)$$

co przez równość współczynników przy jednakowych potęgach z w liczniku i mianowniku daje układ równań

$$\begin{aligned} A + B &= 0, & Av + Bu &= -1, \\ u + v &= 1, & uv &= -1. \end{aligned} \quad (1.17)$$

Dwa równania z dolnego wiersza możemy rozwiązać ze względu na u , co prowadzi do

$$v = 1 - u, \quad u^2 - u - 1 = 0, \quad (1.18)$$

a zatem $u = (1 \pm \sqrt{5})/2$ oraz $v = (1 \mp \sqrt{5})/2$ (górnne i dolne znaki są tu skorelowane). Bez straty ogólności możemy przyjąć

$$u = \frac{1 + \sqrt{5}}{2}, \quad v = \frac{1 - \sqrt{5}}{2}.$$

Druga możliwość doboru znaków jest bowiem równoważna zamianie u z v i A z B , co nie zmienia postaci rozkładu.

Pojawiające się tu liczby są tak fundamentalne, o czym nieco dalej, że zasłużyły na swoje oznaczenia:

$$\phi = \frac{1 + \sqrt{5}}{2}, \quad \hat{\phi} = \frac{1 - \sqrt{5}}{2}. \quad (1.19)$$

Liczبę ϕ nazywamy *złotym podziałem*. Bardzo użyteczne wzory to

$$\phi + \hat{\phi} = 1, \quad \phi - \hat{\phi} = \sqrt{5}, \quad \phi\hat{\phi} = -1. \quad (1.20)$$

Teraz wracamy do dwóch pierwszych równań (1.17) i znajdujemy

$$A = \frac{1}{\sqrt{5}}, \quad B = -\frac{1}{\sqrt{5}}.$$

Wreszcie mamy pożądany rozkład funkcji tworzącej na ułamki proste:

$$f(z) = \frac{1}{\sqrt{5}} \left(\frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z} \right). \quad (1.21)$$

Co uzyskaliśmy? Otóż wzór (1.21) możemy bardzo łatwo rozwinać w szereg potęgowy, ponieważ dla szeregów geometrycznych mamy

$$\begin{aligned} \frac{1}{1 - \phi z} &= 1 + \phi z + \phi^2 z^2 + \phi^3 z^3 + \dots = \sum_{n=1}^{\infty} (\phi z)^n, \\ \frac{1}{1 - \hat{\phi} z} &= 1 + \hat{\phi} z + \hat{\phi}^2 z^2 + \hat{\phi}^3 z^3 + \dots = \sum_{n=1}^{\infty} (\hat{\phi} z)^n. \end{aligned}$$

W efekcie

$$f(z) = \frac{1}{\sqrt{5}} \sum_{n=1}^{\infty} (\phi^n - \hat{\phi}^n) z^n, \quad (1.22)$$

a przyrównanie do (1.10) daje szukaną ogólną postać wyrazów ciągu Fibonacciego:

$$F_n = \frac{1}{\sqrt{5}} (\phi^n - \hat{\phi}^n). \quad (1.23)$$

Pokontemplujmy to równanie. Liczby Fibonacciego są naturalne, a wprowadzone ϕ i $\hat{\phi}$ zawierają $\sqrt{5}$. Jednak liczba $\sqrt{5}$ nie występuje we wzorze (1.23), a to dlatego, że jej nieparzyste potęgi się kasują. Istotnie, korzystając ze wzoru dwumianowego Newtona⁸

$$(x + y)^n = \sum_{k=0}^n \binom{n}{k} x^k y^{n-k}, \quad \binom{n}{k} = \frac{n!}{k!(n-k)!}, \quad (1.24)$$

⁸ Sir Isaac Newton (1642-1727), wielki angielski fizyk i matematyk, odkrywca zasad dynamiki, prawa powszechnego ciążenia, twórca korpuskularnej teorii światła, jeden z ojców analizy matematycznej.

dstajemy z (1.23) po elementarnych przekształceniach wzór

$$F_n = \frac{1}{2^n} \sum_{k=0}^n \binom{n}{k} \sqrt{5}^{k-1} (1 - (-1)^k) = \frac{1}{2^{n-1}} \sum_{m=0}^{[(n-1)/2]} \binom{n}{2m+1} 5^m, \quad (1.25)$$

gdzie sumowanie przebiega tylko po nieparzystych wskaźnikach $k = 2m + 1$. Widzimy w sposób jawny, że $\sqrt{5}$ nie występuje. Symbol $[.]$ w górnej granicy sumy oznacza część całkowitą liczby (zob. rozdz. 1.8, definicja 1.16).

Powróćmy do wzoru (1.23). Co uzyskaliśmy (poza doskonaleniem zawodowym) w stosunku do pierwotnej definicji rekurencyjnej (1.8)? Wyobraźmy sobie, że z jakiegoś powodu potrzebujemy znać bardzo dalekie wyrazy F_n . Przepis rekurencyjny wymagałby obliczenia kolejno wszystkich poprzednich wyrazów, czyli wykonania n dodawań. Wzór ogólny wymaga wykonania dwóch potęgowania. Co prawda dodawanie jest łatwiejsze niż potęgowanie, ale dla dużych n , zwłaszcza, gdy chodzi nam o przybliżony wynik, naprawdę łatwiej jest stosować wzór ogólny niż rekurencję i zdecydowanie zyskujemy na czasie.

Ponieważ $\phi > 1$ oraz $|\hat{\phi}| < 1$, dla dużych wartości n (asymptotycznych) ciąg Fibonacciego rośnie wykładniczo (podobnie, jak ciąg h_n w problemie Wież Hanoi),

$$F_n \sim \phi^n / \sqrt{5},$$

natomiast stosunek kolejnych wyrazów dąży do ϕ ,

$$\lim_{n \rightarrow \infty} \frac{F_{n+1}}{F_n} = \phi. \quad (1.26)$$

Ciekawą własnością liczb Fibonacciego jest tożsamość Cassiniego⁹:

Tw. 1.3 (tożsamość Cassiniego). *Liczby Fibonacciego spełniają związek*

$$F_{n+1}F_{n-1} - F_n^2 = (-1)^n, \quad n \geq 2.$$

Dowód: Podstawiamy do tezy wzór (1.23) i otrzymujemy

$$\frac{1}{5} \left[(\phi^{n+1} - \hat{\phi}^{n+1})(\phi^{n-1} - \hat{\phi}^{n-1}) - (\phi^n - \hat{\phi}^n)^2 \right] = -\frac{1}{5}(\phi\hat{\phi})^{n-1}(\phi - \hat{\phi})^2 = (-1)^n,$$

gdzie użyliśmy wzorów (1.20). \square

Dowód można też przeprowadzić przez indukcję matematyczną.

Liczby Fibonacciego oraz złoty podział występują powszechnie w przyroście. Pokażemy tu kilka najbardziej znanych przykładów. Pierwszy, to (nieco

⁹ Giovanni Domenico Cassini (1625-1712), włoski astronom i matematyk, odkrywca czterech księżyców Saturna i szczelin w jego pierścieniach.

Tabela 1.2: Króliki Fibonacciego w kolejnych czterotygodniowych okresach

Okręs	Dorosłe	Młode	Liczba wszystkich par
1		▽ ○	1
2	▽ ○		1
3	▽ ○	▽ ○	2
4	▽ ○ ▽ ○	▽ ○	3
5	▽ ○ ▽ ○ ▽ ○	▽ ○ ▽ ○	5
6	▽ ○ ▽ ○ ▽ ○ ▽ ○ ▽ ○	▽ ○ ▽ ○ ▽ ○	8
...
n	liczba wszystkich par w okresie $n - 1$	liczba dorosłych par w okresie $n - 1$, czyli liczba wszystkich par w okresie $n - 2$	suma liczb dorosłych par w okresach $n - 1$ i $n - 2$

nierealistyczny) model rozmnażania się królików, pochodzący od samego Fibonacciego. Powiedzmy, że mamy na początku jedną młodą parkę, która dorosłeje po czterech tygodniach. Jak to u królików, po następnych czterech tygodniach (tyle trwa ciąża u królika) przychodzi na świat para króliczków, które wydorośleją po kolejnych czterech tygodniach. Oczywiście w międzyczasie dorosły samiec z pierwszego pokolenia nie próżnował, więc znów po czterech tygodniach mamy dwie dorosłe pary (początkową i wydoroślałą) oraz nowe młode. Schemat powtarza się, co ukazane jest w tabeli 1.2, prowadząc do rekurencji Fibonacciego. Model nie jest realistyczny dla opisu populacji królików. Okazuje się, że dla krótkich czasów króliki rozmnażają się jeszcze szybciej! Model nie uwzględnia bowiem możliwości rodzenia się większej liczby królików w miocie. Natomiast dla dłuższych czasów nie uwzględniliśmy efektów wysycenia zasobów środowiska (pokarm, terytorium), wymierania. Króliki nie mogą rozmnażać się bez końca.

Innego przykładu dostarcza genealogia pszczół. Okazuje się, że sposób rozmnażania się pszczół w naturalny sposób prowadzi do rekurencji Fibonacciego, co jest przedmiotem ćw. 1.14.

Złoty podział występuje w przyrodzie, sztuce i architekturze, gdzie uznawany jest za idealną proporcję:

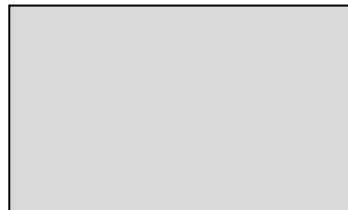
$$\phi = \frac{1 + \sqrt{5}}{2} \simeq 1.61803 \quad (1.27)$$

Zapamiętaj tę liczbę! Najbardziej pospolity przykład tej proporcji to podział odcinka o długości $a + b$ w taki sposób, aby powstały odcinki o długościach a (dłuższy) i b (krótszy) spełniały proporcję

$$\frac{a + b}{a} = \frac{a}{b}, \quad (1.28)$$



Rysunek 1.5: Złoty podział odcinka

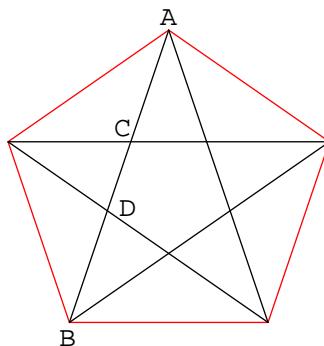


Rysunek 1.6: Złoty prostokąt: stosunek długości boków wynosi ϕ

czyli „całość do dłuższy równa się dłuższy do krótszy” (zob. rys. 1.5). Rozwiążanie daje właśnie $a/b = \phi$. Złoty prostokąt, czyli prostokąt, którego boki spełniają tę proporcję, przedstawiony jest na rys. 1.6. Można argumentować, że prostokąt ten nie jest ani za gruby, ani za chudy, jest taki „najbardziej prostokątny”.

Kolejny przykład z geometrii dotyczy pentagramu, rys. 1.7. Można pokazać (ćw. 1.18), że przekatne tej figury dzielą się w złotej proporcji, $|AB|/|AD| = \phi$, również $|AD|/|AC| = \phi$. Złota liczba ϕ występuje też w architekturze. Stosunek wysokości ściany bocznej do połowy podstawy Wielkiej Piramidy w Gizie wynosi 1.61805 – zgodność na poziomie 10^{-5} z (1.27). Fronton Partenonu wpisany jest w złoty prostokąt itd. Liczba ϕ jest „kultowa” i można się jej doszukiwać w bardzo wielu miejscach (ćw. 1.19 i 1.20). Miejmy tu jednak na uwadze fakt, że odcinków w bardziej złożonych obiektach i ich wzajemnych proporcji jest bardzo dużo i zawsze któraś ma szansę być blisko ϕ .

Bardzo ciekawym i biologicznie uzasadnionym mechanizmem jest powszechnie występowanie liczb Fibonacciego i złotego podziału w filaksji, czyli wzroście roślin. Obszerna i przystępna dyskusja tego zagadnienia znajduje się w [6]. W informatyce (teoria struktur danych) istotne znaczenie praktyczne mają tzw. kopce



Rysunek 1.7: Pentagram i złoty podział: $|AB|/|AD| = \phi = |AD|/|AC|$

Fibonacciego, zob. [4]. Ostatnio magia liczb Fibonacciego i złotego podziału została spopularyzowana przez książkę Dana Browna *Kod Leonarda da Vinci* [36] i oparty na niej film. Pilny student tej części wykładu, rozumiejąc tajniki *Kodu*, wywrze nieodparte wrażenie na osobie zabranej ze sobą do kina!

1.5 Równanie charakterystyczne

W tym podrozdziale poznamy bardzo potężną, a zarazem prostą metodę rozwiązywania rekurencji liniowych, sprowadzającą zagadnienie do procedur tak elementarnych, jak znajdowanie pierwiastków wielomianu oraz rozwiązywanie układu równań liniowych.

Rozważmy na początek nieco ogólniejszy przepis rekurencyjny niż ciąg Fibonacciego, mianowicie

$$a_n = \alpha a_{n-1} + \beta a_{n-2}, \quad (1.29)$$

gdzie α i $\beta \neq 0$ są pewnymi stałymi rzeczywistymi. Ustalamy też wartości dwóch dowolnych (niekoniecznie dwóch pierwszych) wyrazów ciągu, a_l i a_m , $l \neq m$ – warunki te nazywamy *warunkami początkowymi*. Oczywiście, równanie (1.29) możemy zapisać w postaci

$$a_n - \alpha a_{n-1} - \beta a_{n-2} = 0. \quad (1.30)$$

Równaniem *charakterystycznym* ciągu (1.29) nazywamy równanie postaci

$$u^2 - \alpha u - \beta = 0, \quad (1.31)$$

gdzie w ogólności u jest liczbą zespoloną. Porównując (1.30) i (1.31), widzimy, jak powstaje równanie charakterystyczne: w miejsce a_n wpisujemy u^2 , w miejsce a_{n-1} wstawiamy u , a w miejsce a_{n-2} dajemy $u^0 = 1$. Obniżeniu wskaźnika n odpowiada obniżenie potęgi u . Stałe α i β przepisujemy.

A teraz podstawowe twierdzenie dla rekurencji (1.29):

Tw. 1.4.

- Jeśli równanie charakterystyczne ma dwa różne (w ogólności zespolone) pierwiastki u_1 i u_2 , to

$$a_n = c_1 u_1^n + c_2 u_2^n, \quad (1.32)$$

- jeśli ma jeden podwójny pierwiastek u_0 , to

$$a_n = (c_1 + c_2 n) u_0^n. \quad (1.33)$$

Stałe c_1 i c_2 dobieramy w taki sposób, aby odtworzyć zadane wyrazy a_l i a_m , $l \neq m$.

Dowód: Rozważmy najpierw przypadek różnych pierwiastków, $u_1 \neq u_2$. Wzór (1.32) dla $n = l$ i $n = m$ daje następujący układ równań liniowych na współczynniki c_1 i c_2 :

$$\begin{aligned} a_l &= c_1 u_1^l + c_2 u_2^l, \\ a_m &= c_1 u_1^m + c_2 u_2^m. \end{aligned} \quad (1.34)$$

Ponieważ $\beta \neq 0$, pierwiastki u_1, u_2 są różne od zera. Wyznacznik układu (1.34) wynosi

$$u_1^l u_2^m - u_1^m u_2^l = u_1^l u_2^m \left(1 - (u_1/u_2)^{m-l}\right).$$

Ponieważ $u_1 \neq u_2$, wyznacznik ten jest różny od zera, czyli układ (1.34) jest układem Cramera i na mocy twierdzenia z algebra ma rozwiązanie dla dowolnych zadanych wartości a_l i a_m . Następnie musimy sprawdzić, czy rozwiązanie (1.32) spełnia rekurencję (1.29). Podstawiając, otrzymujemy w istocie

$$a_n = c_1 u_1^n + c_2 u_2^n = c_1 (\alpha u_1^{n-1} + \beta u_1^{n-2}) + c_2 (\alpha u_2^{n-1} + \beta u_2^{n-2}) = \alpha a_{n-1} + \beta a_{n-2},$$

gdzie w drugiej równości wykorzystaliśmy fakt, że u_1 i u_2 spełniają równanie (1.31).

Teraz rozważmy tzw. przypadek zdegenerowany, czyli przypadek równych pierwiastków $u_1 = u_2 = u_0$. Zauważmy, że wówczas

$$\begin{aligned} \alpha^2 &= 4\beta, \\ u_0 &= \alpha/2. \end{aligned} \quad (1.35)$$

Warunek $\beta \neq 0$ implikuje $\alpha \neq 0$ i $u_0 \neq 0$. Układ równań na współczynniki ma postać

$$\begin{aligned} a_l &= (c_1 + lc_2)u_0^l, \\ a_m &= (c_1 + mc_2)u_0^m, \end{aligned} \quad (1.36)$$

a jego rozwiązanie jest postaci

$$\begin{aligned} c_1 &= \frac{ma_l u_0^{-l} - la_m u_0^{-m}}{m-l}, \\ c_2 &= \frac{a_m u_0^{-m} - a_l u_0^{-l}}{m-l}. \end{aligned} \quad (1.37)$$

Pozostaje sprawdzić, że rozwiązanie (1.33) z wartościami współczynników (1.37) spełnia rekurencję (1.29). Otrzymujemy

$$\begin{aligned} a_n - \alpha a_{n-1} - \beta a_{n-2} &= (c_1 + c_2 n)u_0^n - \alpha(c_1 + c_2(n-1))u_0^{n-1} - \beta(c_1 + c_2(n-2))u_0^{n-2} \\ &= (c_1 + nc_2)(u_0^n - \alpha u_0^{n-1} - \beta u_0^{n-2}) + \alpha u_0^{n-1} + 2\beta u_0^{n-2} = 0 + u_0^{n-2}(\alpha u_0 + 2\beta) \\ &= u_0^{n-2}(\alpha^2/2 - \alpha^2/2) = 0, \end{aligned}$$

gdzie skorzystaliśmy z tożsamości (1.35). \square

Algorytm rozwiązywania (jednorodnej) rekurencji liniowej rzędu 2 jest następujący:

Alg. 1.2 (rozwiązywanie jednorodnej rekurencji liniowej rzędu 2).

1. Napisz stosowne równanie charakterystyczne i znajdź jego (w ogólności zespółonej) pierwiastki.
2. W zależności od tego, czy pierwiastki są sobie równe, czy nie, napisz odpowiedni wzór ogólny na a_n zawierający stałe c_1 i c_2 .
3. Korzystając z warunków początkowych, napisz i rozwiąż układ dwóch równań liniowych na stałe c_1 i c_2 .
4. Sprawdź otrzymany wynik dla kilku pierwszych wyrazów ciągu.

Jako ilustrację metody równania charakterystycznego weźmy ponownie ciąg Fibonacciego (1.8). W tym przypadku $\alpha = \beta = 1$, zatem równanie charakterystyczne ma postać $u^2 - u - 1 = 0$, jego pierwiastki to $u_1 = \phi$, $u_2 = \hat{\phi}$, a na mocy (1.32) $a_n = c_1\phi^n + c_2\hat{\phi}^n$. Stałe c_1 i c_2 wyznaczamy z warunków początkowych, dających liniowy układ dwóch równań o dwóch niewiadomych:

$$\begin{aligned} 1 &= a_1 = c_1 \frac{1 + \sqrt{5}}{2} + c_2 \frac{1 - \sqrt{5}}{2}, \\ 1 &= a_2 = c_1 \left(\frac{1 + \sqrt{5}}{2} \right)^2 + c_2 \left(\frac{1 - \sqrt{5}}{2} \right)^2. \end{aligned}$$

Rozwiązanie to $c_1 = 1/\sqrt{5}$, $c_2 = -1/\sqrt{5}$, więc ostatecznie otrzymujemy ponownie wzór (1.23). Zauważmy, że metoda równania charakterystycznego wymagała mniejszej pracy niż zastosowanie funkcji tworzącej: musielibyśmy rozwiązać równanie kwadratowe oraz układ dwóch równań liniowych.

Rekurencja rozważana w Tw. 1.4 była rzędu drugiego, tzn. n -ty wyraz zależał od dwóch wyrazów bezpośrednio go poprzedzających. Rozważymy teraz ogólny przypadek rekurencji liniowej jednorodnej rzędu r , przy czym $r \geq 1$.

Def. 1.10. *Rekurencją liniową jednorodną rzędu r o stałych współczynnikach nazywamy rekurencję postaci*

$$a_n = \alpha_1 a_{n-1} + \alpha_2 a_{n-2} + \cdots + \alpha_r a_{n-r},$$

gdzie $\alpha_r \neq 0$ oraz stałe α_i ($i = 1, \dots, r$) nie zależą od n . Przepis na a_n zależy więc teraz od r poprzednich wyrazów. Określenie tej rekurencji wymaga zadania r wyrazów (warunki początkowe). Równaniem charakterystycznym jest teraz równanie wielomianowe stopnia r ,

$$u^r = \alpha_1 u^{r-1} + \alpha_2 u^{r-2} + \cdots + \alpha_r u^0.$$

Określenie *liniowa* oznacza tu tyle, że po prawej stronie występują pierwsze potęgi a_{n-1}, \dots, a_{n-r} , a nie np. ich kwadraty. Słowo *jednorodna* oznacza, że po prawej stronie mamy tylko $\alpha_1 a_{n-1} + \alpha_2 a_{n-2} + \dots + \alpha_r a_{n-r}$, bez dodatkowego członu niezależnego od wyrazów a_k (patrz podrozdział 1.6). Poniższe twierdzenie podaje ogólny przepis rozwiązywania rekurencji liniowej jednorodnej rzędu r .

Tw. 1.5. *Na mocy podstawowego twierdzenia algebry równanie charakterystyczne posiada s pierwiastków, w ogólności zespolonych, u_1, u_2, \dots, u_s , o odpowiednich krotnościach k_1, k_2, \dots, k_s , przy czym $k_1 + k_2 + \dots + k_s = r$. Rozwiązanie rekurencji (1.10) ma postać*

$$\begin{aligned} a_n &= (c_{1,1} + c_{1,2}n + c_{1,3}n^2 + \dots + c_{1,k_1}n^{k_1-1})u_1^n \\ &\quad + (c_{2,1} + c_{2,2}n + c_{2,3}n^2 + \dots + c_{2,k_2}n^{k_2-1})u_2^n \\ &\quad + \dots \\ &\quad + (c_{s,1} + c_{s,2}n + c_{s,3}n^2 + \dots + c_{s,k_s}n^{k_s-1})u_s^n. \end{aligned}$$

Współczynniki $c_{i,j}$, których jest w sumie r , wyznaczamy, rozwiązyując układ równań liniowych otrzymany za pomocą warunków początkowych. W ogólności, jeśli pierwiastki u_i są zespolone, stałe $c_{i,j}$ są zespolone.

Notacja się nam tutaj, ze względem na ogólny charakter rozwiązania, nieco skomplikowała, ale istota pozostaje prosta. Rozwiązanie dla a_n jest sumą po wszystkich pierwiastkach zawierającą u_i^n . Jeśli dany pierwiastek jest wielokrotny, odpowiadający mu człon jest dodatkowo przemnożony przez wielomian stopnia „krotność minus jeden”. Jeśli pierwiastek jest jednokrotny, to wielomian ten staje się współczynnikiem. Poniższe przykłady i ćwiczenia na końcu rozdziału pozwolą przełożyć ogólny zapis twierdzenia na konkretne przypadki.

Dowód Tw. 1.5, będący uogólnieniem dowodu Tw. 1.4, pomijamy. Twierdzenie pokazuje, że rozwiązanie rekurencji jest równoważne rozwiązaniu równania charakterystycznego, co zawsze możemy zrobić analitycznie¹⁰ dla $r \leq 4$.

Rozwiązanie, w którym nie określamy stałych $c_{i,j}$, nazywamy *rozwiązaniem ogólnym* rekurencji 1.10. Konkretnie wartości tych stałych zależą rzecz jasna od warunków początkowych. Tak więc rozwiązanie ogólne to postać rozwiązania słuszna dla dowolnych warunków początkowych.

A teraz prosty wniosek dotyczący zachowania a_n przy $n \rightarrow \infty$:

Wniosek 1.2. *Zachowanie a_n dla dużych wartości n zależy od pierwiastka u_{\max} , który ma największą wartość bezwzględną. W szczególności*

$$a_n \sim n^{k_{\max}-1} u_{\max}^n,$$

gdzie k_{\max} jest krotnością pierwiastka u_{\max} , natomiast $\lim_{n \rightarrow \infty} a_n/a_{n-1} = u_{\max}$.

¹⁰ Jak wynika z teorii Galois w algebrze, dla $r > 4$ w ogólnym przypadku nie istnieje metoda znalezienia w dokładny sposób pierwiastków wielomianu stopnia r . Możemy to jednak zawsze zrobić numerycznie z dowolną dokładnością.

Jako prosty formalny przykład zastosowania Tw. 1.5 rozważmy rekurencję rzędu trzeciego

$$\begin{aligned} a_n &= a_{n-1} - a_{n-2} + a_{n-3}, \\ a_0 &= 0, \quad a_1 = 1, \quad a_2 = 2, \end{aligned} \tag{1.38}$$

o równaniu charakterystycznym $u^3 - u^2 + u - 1 = (u-1)(u^2+1) = 0$. Mamy więc trzy pierwiastki jednokrotne: $u_1 = 1$, $u_2 = i$, $u_3 = -i$, wobec czego rozwiązań ogólnego ma postać $a_n = c_1 + c_2 i^n + c_3 (-i)^n$. Układ równań liniowych

$$\begin{aligned} a_0 &= 0 = c_1 + c_2 + c_3, \\ a_1 &= 1 = c_1 + c_2 i - c_3 i, \\ a_2 &= 2 = c_1 - c_2 - c_3, \end{aligned} \tag{1.39}$$

ma rozwiązanie $c_1 = 1$, $c_2 = c_3 = -1/2$, a więc

$$a_n = 1 - \frac{1}{2}i^n - \frac{1}{2}(-i)^n.$$

Ciąg jest okresowy: $(a_n) = 0, 1, 2, 1, 0, 1, 2, 1, \dots$. Mimo faktu, że pierwiastki x_2 i x_3 są zespolone, wszystkie wyrazy ciągu są rzeczywiste. Co więcej, są całkowite, co w oczywisty sposób wynika z przepisu rekurencyjnego (1.38) – początkowe wyrazy ciągu są całkowite, a kolejne są ich kombinacją liniową z całkowitymi współczynnikami, więc rzecz jasna muszą też być całkowite. Wszystko się zgadza!

1.6 Rekurencja liniowa niejednorodna

Rozważany w podrozdziale 1.3 przykład liczb trójkątnych różni się jakościowo tym od rekurencji Hanoi czy Fibonacciego, że funkcja F z def. 1.1 zawiera składnik niezależny od wyrazów ciągu. Mieliśmy bowiem $F(p_n; n) = p_n + n$, por. (1.5). Pierwszy człon, p_n , nazywamy jednorodnym, a drugi, w tym przypadku n , *niejednorodnym*. W ogólnej sytuacji mamy następującą definicję:

Def. 1.11. *Rekurencję liniową niejednorodną rzędu r nazywamy rekurencję postaci*

$$a_n = \alpha_1 a_{n-1} + \alpha_2 a_{n-2} + \cdots + \alpha_r a_{n-r} + g(n), \tag{1.40}$$

gdzie $\alpha_r \neq 0$, a $g(n)$ jest dowolną funkcją zmiennej n niezależną od wyrazów a_i . Określenie rekurencji wymaga zadania r wyrazów (warunki początkowe).

Po prawej stronie mamy więc człony jednorodne $\alpha_1 a_{n-1} + \alpha_2 a_{n-2} + \cdots + \alpha_r a_{n-r}$ oraz człon *niejednorodny* $g(n)$. Rozwiązywanie rekurencji z definicji 1.11 jest bardziej skomplikowane od przypadku rekurencji jednorodnej 1.10 i daje się ogólnie przeprowadzić tylko dla pewnych postaci członu niejednorodnego $g(n)$. Metoda postępowania zawarta jest w poniższych definicjach i twierdzeniach.

Def. 1.12. *Rekurencją jednorodną stwarzyszoną z 1.11 nazywamy rekurencję postaci 1.11 z opuszczonym członem $g(n)$, tj.*

$$a_n = \alpha_1 a_{n-1} + \alpha_2 a_{n-2} + \cdots + \alpha_r a_{n-r}.$$

Def. 1.13. *Rozwiązańiem szczególnym rekurencji 1.11 nazywamy pewne dowolne rozwiązanie równania (1.40), niekoniecznie spełniające warunki początkowe.*

Tw. 1.6. *Rozwiązańem ogólnego rekurencji 1.11 jest suma jej rozwiązania szczególnego, oznaczonego jako s_n , oraz rozwiązania ogólnego stwarzyszonej rekurencji jednorodnej, $j(n)$. Suma ta, $j(n) + s(n)$, musi spełniać warunki początkowe.*

Dowód: Dowód przebiega przez sprawdzenie. Wstawiamy postać $a_n = s_n + j_n$ do (1.40), co daje

$$\begin{aligned} s_n + j_n &= \alpha_1(s_{n-1} + j_{n-1}) + \cdots + \alpha_r(s_{n-r} + j_{n-r}) + g(n) \\ &= \alpha_1 s_{n-1} + \cdots + \alpha_r s_{n-r} + g(n) + \alpha_1 j_{n-1} + \cdots + \alpha_r j_{n-r}. \end{aligned}$$

W drugiej linijce rozpoznajemy dodane stronami równanie (1.40), spełnione z definicji dla rozwiązania szczególnego s_n oraz stwarzyszone równanie jednorodne, spełnione dla j_n . \square

Rozwiązywanie rekurencji 1.11 sprowadza się więc do znalezienia s_n i j_n . Następnie, tak samo jak w przypadku rekurencji jednorodnej, wyznaczamy stałe $c_{i,j}$ występujące w j_n . Znalezienie rozwiązania ogólnego rekurencji jednorodnej j_n jest zadaniem prostym: postępujemy według przepisu z Tw. 1.5. Natomiast znalezienie szczególnego rozwiązania rekurencji niejednorodnej s_n nie zawsze jest proste. Wykonalność zależy od konkretnej postaci członu niejednorodnego $g(n)$. Możemy znaleźć rozwiązanie s_n dla często spotykanego przypadku, kiedy $g(n)$ ma postać wielomian razy n -ta potęga dowolnej stałej:

$$g(n) = (d_0 + d_1 n + d_2 n^2 + \cdots + d_m n^m) t^n, \quad (1.41)$$

gdzie d_i , $i = 0, \dots, m$, oraz t są pewnymi stałymi (niezależnymi od n). Mamy teraz dwa przypadki: albo t jest jednym z pierwiastków równania charakterystycznego stwarzyszonej rekurencji jednorodnej, albo nie jest. Poniższe twierdzenie rozstrzyga, jak postępować w obu przypadkach.

Tw. 1.7. *Rekurencja 1.11 z członem niejednorodnym (1.41) ma rozwiązanie szczególne postaci:*

1. $s_n = (w_0 + w_1 n + \cdots + w_m n^m) t^n$, jeśli t nie jest pierwiastkiem równania charakterystycznego,
2. $s_n = n^k (w_0 + w_1 n + \cdots + w_m n^m) t^n$, jeśli t jest pierwiastkiem równania charakterystycznego. W tym przypadku k jest krotnością tego pierwiastka.

W obu przypadkach współczynniki w_i , $i = 0, \dots, m$, wyznaczamy poprzez wstawienie ogólnej postaci rozwiązania do równania (1.40) i porównania współczynników przy kolejnych potęgach n .

Dowód pomijamy¹¹.

Wniosek 1.3. Wynika stąd, że asymptotycznie zachowanie rozwiązania rekurencji niejednorodnej jest określone przez liczbę o największej wartości bezwzględnej spośród zbioru $\{u_1, \dots, u_s, t\}$. Mamy następujące przypadki:

1. Największą wartość bezwzględną ma pewien pierwiastek u_{\max} o krotności k_{\max} , przy czym $u_{\max} \neq t$:

$$a_n \sim n^{k_{\max}-1} u_{\max}^n.$$

2. Największą wartość bezwzględną ma t niebędące pierwiastkiem równania charakterystycznego:

$$a_n \sim n^m t^n.$$

3. Największą wartość bezwzględną ma pewien pierwiastek u_{\max} o krotności k_{\max} , przy czym $t = u_{\max}$:

$$a_n \sim n^{m+k_{\max}} t^n.$$

Podsumujmy podane wcześniej przepisy:

Alg. 1.3 (rozwiązywanie jednorodnej rekurencji liniowej rzędu n).

1. Napisz stosowne równanie charakterystyczne, będące wielomianem stopnia n i znajdź jego pierwiastki. Pierwiastki są w ogólności zespolone.
2. Napisz odpowiedni wzór ogólny (zależny od krotności pierwiastków) na a_n zawierający n stałych $c_{i,j}$.
3. Wykorzystując n warunków początkowych, napisz i rozwiąż układ n równań liniowych na te stałe.
4. Sprawdź otrzymany wynik dla kilku pierwszych wyrazów ciągu.

Alg. 1.4 (rozwiązywanie niejednorodnej rekurencji liniowej rzędu n).

1. Opuść człon niejednorodny i wykonaj dwa pierwsze kroki algorytmu 1.3, co daje ogólne rozwiązanie rekurencji jednorodnej j_n .
2. Przy użyciu dowolnej metody znajdź szczegółowe rozwiązanie rekurencji niejednorodnej s_n (wykonalność tego zadania zależy od postaci członu niejednorodnego $g(n)$). Dla $g(n)$ postaci (1.41) postępuj zgodnie z Tw. 1.7.

¹¹ Obyty czytelnik zauważtu tu podobieństwo do tzw. metody przewidywań rozwiązywania układów niejednorodnych liniowych równań różniczkowych o stałych współczynnikach.

3. Zsumuj otrzymane rozwiązania, uzyskując ogólne rozwiązanie rekurencji niejednorodnej, $a_n = j_n + s_n$, zależne od n stałych $c_{i,j}$.
4. Korzystając z warunków początkowych, napisz i rozwiąż układ n równań liniowych na te stałe.
5. Sprawdź otrzymany wynik dla kilku pierwszych wyrazów ciągu.

Podawszy całą „ciężka artylerię” rozwiązywania rekurencji liniowych, czas na przykłady. Ich prześledzenie jest niezbędne celem oswojenia się z ogólną notacją powyższych procedur. Zaczniemy od ponownego rozwiązania problemu liczb trójkątnych. Jak pamiętamy, rekurencja ma postać

$$p_n = p_{n-1} + n, \quad (1.42)$$

zatem rząd rekurencji $r = 1$, a człon niejednorodny wynosi $g(n) = n$. Stowarzyszona rekurencja jednorodna to $j_n = j_{n-1}$, a jej równanie charakterystyczne ma postać $u = 1$, mamy zatem jeden pierwiastek $u_1 = 1$ o krotności $k = 1$. Na mocy Tw. 1.5 rozwiązanie ogólne ma zatem postać $j_n = c_1 1^n = c_1$. Teraz znajdziemy rozwiązanie szczególne rekurencji niejednorodnej (1.42). Ponieważ $g(n) = n$, stałe z równania (1.41) wynoszą $d_0 = 0$, $d_1 = 1$, $m = 1$ oraz $t = 1$. Ponieważ t ma tę samą wartość co pierwiastek u_1 , mamy przypadek 2 z Tw. 1.7. Bierzemy więc

$$s(n) = n^1(w_0 + w_1 n)1^n = w_0 n + w_1 n^2$$

i podstawiamy do równania (1.42), co daje

$$w_0 n + w_1 n^2 = w_0(n-1) + w_1(n-1)^2 + n,$$

a po rozwinięciu i uproszczeniu

$$0 = -w_0 + w_1 - 2w_1 n + n.$$

Równanie to musi zachodzić dla wszystkich n , a więc współczynniki przy poszczególnych potęgach n po prawej stronie równania muszą znikać, co daje układ równań

$$\begin{aligned} -w_0 + w_1 &= 0 \\ -2w_1 + 1 &= 0, \end{aligned}$$

skąd $w_0 = w_1 = 1/2$. Tak więc ogólne rozwiązanie rekurencji (1.42) ma postać

$$p_n = j_n + s_n = c_1 + \frac{n(n+1)}{2}.$$

Pozostaje wyznaczenie stałej c_1 za pomocą warunku początkowego $p_1 = 1$. Wstawnienie do powyższego równania $n = 1$ daje $1 = c_1 + 1$, skąd $c_1 = 0$ i ostatecznie $p_n = n(n+1)/2$, w zgodzie z wcześniejszym wynikiem (1.7).

Drugi przykład to modyfikacja ciągu Fibonacciego:

$$a_n = a_{n-1} + a_{n-2} + 3^n, \quad a_1 = a_2 = 1. \quad (1.43)$$

Problem jednorodny rozważaliśmy już wcześniej, zatem $j_n = c_1\phi^n + c_2\hat{\phi}^n$, gdzie złoty podział ϕ i liczba $\hat{\phi}$ są zdefiniowane w (1.19). Człon niejednorodny ma postać $g(n) = 3^n$, skąd wyczytujemy $m = 0$, $t = 3$. Ponieważ t nie jest równe żadnemu z pierwiastków równania charakterystycznego, mamy przypadek 1 z Tw. 1.7, a rozwiązanie szczególne równania niejednorodnego ma postać $s_n = w_03^n$. Wstawienie do równania (1.43) daje $w_03^n = w_03^{n-1} + w_03^{n-2} + 3^n$, skąd po wymnożeniu obu stron przez 3^{2-n} dostajemy $w_03^2 = w_03 + w_0 + 3^2$, zatem $w_0 = 9/5$. Mamy więc $a_n = j_n + s_n = c_1\phi^n + c_2\hat{\phi}^n + (9/5)3^n$. Uwzględnienie warunków początkowych (1.43) daje układ równań na c_1 i c_2 , który ma rozwiązanie $c_1 = (-27 + \sqrt{5})/5$, $c_2 = (-27 - \sqrt{5})/5$. Ostatecznie

$$a_n = \frac{1}{5} \left(3^{n+2} + (-27 + \sqrt{5})\phi^n - (27 + \sqrt{5})\hat{\phi}^n \right).$$

Kilka pierwszych wyrazów tego ciągu to

$$1, 1, 29, 111, 383, 1223, 3793, 11577, 35053, 105679, \dots,$$

skąd łatwo sprawdzić, że rzeczywiście spełniona jest rekurencja (1.43).

Zauważmy jeszcze, że asymptotycznie $a_n \sim 3^n$, a nie jak ϕ^n . Jest to zgodne z ogólnym prawidłem wynikającym z Tw. 1.6 i 1.7. Dla dużych n zachowanie ciągu dane jest przez liczbę o największej wartości bezwzględnej wybranej spośród pierwiastków równania charakterystycznego (tutaj ϕ i $\hat{\phi}$) oraz liczby t (tutaj $t = 3$). W obecnym przypadku t „wygrywa”, stąd asymptotyka $a_n \sim 3^n$.

1.7 Ruina gracza

W niniejszym podrozdziale zilustrujemy poznane powyżej metody rozwiązywania rekurencji kolejnym słynnym problemem, a mianowicie zadaniem o *ruinie gracza* [37]. Problem ten jest ciekawy i kształcający, zawiera w sobie bowiem elementy poznanej właśnie teorii rekurencji, a także rachunku prawdopodobieństwa, teorii błądzenia przypadkowego (procesy Markowa¹²), teorii równań liniowych, wreszcie teorii grafów i automatów skończonych. Podrozdział ten może być również przeczytany w innej kolejności, tj. po zapoznaniu się z rozdz. 3 i 5.

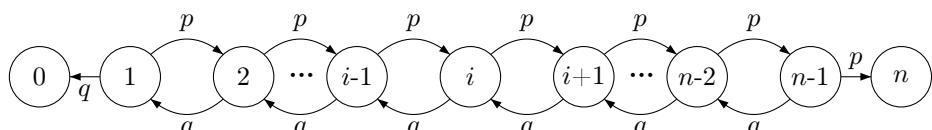
Zagadnienie jest następujące: hazardzista rozpoczyna grę w kasynie, posiadając pewną liczbę i jednakowych monet. Powiedzmy, że gra w oko. Gra ma takie reguły, że w kolejnych rozdaniach z prawdopodobieństwem p gracz wygrywa – wtedy stan jego posiadania wzrasta o jedną monetę, lub z prawdopodobieństwem q przegrywa – wówczas traci jedną monetę. Oczywiście $p + q = 1$, bo musi albo

¹² Andriej Andriejewicz Markow (1856-1922), rosyjski matematyk, rozwinał teorię procesów stochastycznych.

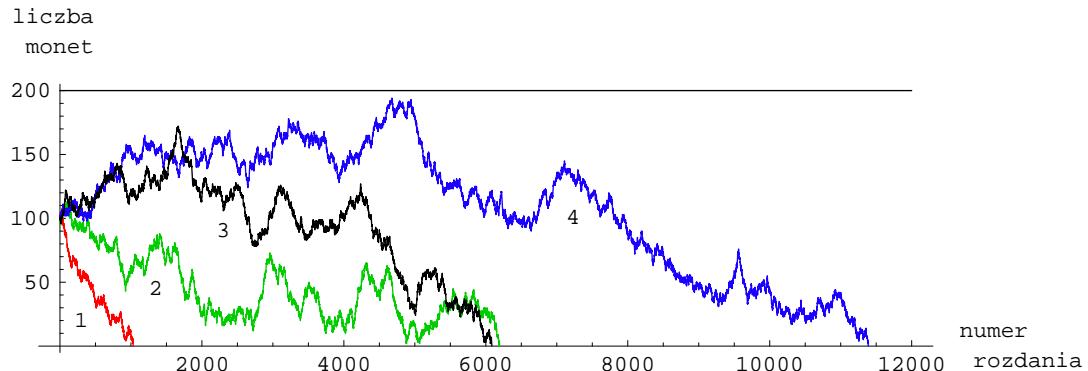
wygrać, albo przegrać. Rozdania są powtarzane „do skutku”. Jeśli hazardzista straci wszystko, to bankrutowie. Jeśli natomiast jego stan posiadania wzrośnie do n monet, to rozbija bank i szczęśliwy opuszcza kasyno.

Sytuację przedstawia rys. 1.8. Mamy tu do czynienia z tzw. *skończonym automatem probabilistycznym* (automaty opisane są bardziej szczegółowo w rozdz. 6). *Skończony* odnosi się do skończonej liczby stanów, *probabilistyczny* oznacza, że odbywają się losowania dotyczące przejść ze stanu do stanu. Gracz rozpoczyna grę ze stanu początkowego i , odbywa się losowanie (rozdanie) i gracz z prawdopodobieństwem p przesuwa się do stanu $i + 1$ lub z prawdopodobieństwem q do stanu $i - 1$ itd. Automat ma dwa *stany akceptujące*, 0 i n . Stan 0 oznacza „spłukanie”, a stan n rozbicie banku – w obu przypadkach gra się kończy. Zauważmy, że ze stanów akceptujących „nie ma odwrotu”, więc na diagramie nie wychodzą z nich żadne strzałki. Proces całej gry polega więc na przypadkowym błędzeniu po diagramie 1.8, aż do znalezienia się w którymś z dwóch stanów akceptujących.

Na początek pokibicujmy grze. Przykładowa historia dla czterech hazardistów przedstawiona jest na rys. 1.9. Wykres przedstawia liczbę monet (osi pionowa) będących w posiadaniu gracza po kolejnych rozdaniach (osi pozioma). Przyjelismy realistyczną sytuację gry nieco niesprawiedliwej, $p = 0.49$, $q = 0.51$. Co widzimy? Dla każdego z graczy liczba posiadanych monet *fluktuuje* w bardzo przypadkowy sposób (przypominający notowania kursów walut lub spółek giełdowych). Ponieważ $q > p$, po dostatecznie długim graniu liczba monet u każdego z czterech graczy maleje do zera i wszyscy zostają zrujnowani. Co prawda gracz 4 miał początkowo dość dużo szczęścia i omal nie rozbił banku po ok. 5000 rozdań (gdyby wtedy wyszedł z kasyna, to wygrałby netto niemal 100 monet!), jednak potem prawa statystyki zadziałyły nieubłaganie i nasz hazardzista zgrał się do cna. Gracze 2 i 3 opuścili kasyno po ok. 6000 rozdań, gracz 1 miał jeszcze mniej szczęścia, kończąc po ok. 1000 rozdań. Zauważmy też, że przy przyjętych wartościach p , q , początkowej liczbie monet (100) i stawce (jedna moneta) gracze grają bardzo długo, pierwszy ok. 1000, drugi i trzeci ok. 6000, a czwarty aż ok. 11500 rozdań. Rozrzut czasu gry jest bardzo duży. Fascynujące jest to, że tak



Rysunek 1.8: Diagram automatu probabilistycznego dla ruinę gracza. Etykietą stanu, oznanego kółkiem, jest liczba monet będąca aktualnie w posiadaniu gracza. Kropki oznaczają dodatkowe, nienarysowane stany. Etykietę p i q przy strzałkach oznaczają prawdopodobieństwa przejścia między stanami: gracz wygrywa pojedynczą grę z prawdopodobieństwem p i przegrywa z prawdopodobieństwem q . Gracz zaczyna ze stanu i , po czym „błądzi przypadkowo” po diagramie aż do momentu, gdy przegra wszystko (stan 0) lub rozbije bank (stan n)



Rysunek 1.9: Historia pobytu czterech graczy w kasynie, czyli wykres stanu posiadania w kolejnych rozdaniach ($p = 0.49$, $q = 0.51$)

prosta reguła, jak wielokrotne losowanie, prowadzi do nader skomplikowanych przebiegów z rys. 1.9.

Zajmiemy się teraz ilościowym opisem ruiny gracza. Pierwsze pytanie, to jakie jest prawdopodobieństwo rozbicia banku. Oznaczmy prawdopodobieństwo uzyskania n monet dla gracza posiadającego (w danej chwili) k monet jako a_k , $0 \leq k \leq n$. Wiemy, że $a_0 = 0$, bo jeśli gracz nie ma już monet, to jako bankrut nie może dalej grać i stracił możliwość rozbicia banku. Z drugiej strony, $a_n = 1$, bo jeśli ma n monet, to właśnie rozbil bank, a prawdopodobieństwo zdarzenia pod warunkiem, że zdarzenie zaszło, jest 1. W kolejnych rozdaniach gracz albo z prawdopodobieństwem p wygrywa, wtedy prawdopodobieństwo rozbicia banku wynosi a_{k+1} , albo z prawdopodobieństwem q przegrywa, wówczas jego szansa na zwycięstwo wynosi a_{k-1} . Stan, w którym gracz posiada k monet, możemy w skrócie nazywać stanem k . W trakcie gry gracz błędzi przypadkowo między kolejnymi stanami k . Teraz skorzystajmy ze wzoru na tzw. prawdopodobieństwo całkowite, co jest kluczowe w naszej analizie. Napiszmy najpierw słownie następujący fakt:

Prawdopodobieństwo rozbicia banku ze stanu k jest równe prawdopodobieństwu przejścia do stanu $k + 1$ razy prawdopodobieństwo rozbicia banku ze stanu $k + 1$, plus prawdopodobieństwo przejścia do stanu $k - 1$ razy prawdopodobieństwo rozbicia banku ze stanu $k - 1$.

Dlaczego tak jest? Rozważmy N graczy (N jest bardzo duże, rozumiemy, że dąży do nieskończoności) startujących ze stanu k . Gracze grają „do spodu” i część z nich, W , wygrywa, a część przegrywa. Uprzedzając dyskusję z rozdz. 3, przypomnijmy, że w naszym przypadku prawdopodobieństwo rozbicia banku obliczamy (z definicji) jako stosunek liczby graczy, którzy wygrali do liczby wszystkich graczy, czyli

$$a_k = W/N.$$

A teraz alternatywny sposób obliczenia a_k : po pierwszym rozdaniu N^+ graczy wygrywa jedną monetę, czyli przechodzi do stanu $k+1$, natomiast N^- przegrywa i przechodzi do stanu $k-1$. Z definicji prawdopodobieństwa dla tych przejść mamy $N^+/N = p$ i $N^-/N = q$. Z N^+ graczy będących w stanie $k+1$ liczba W^+ wygrywa, zatem $a_{k+1} = W^+/N^+$, podobnie $a_{k-1} = W^-/N^-$. Łącząc w elementarny sposób otrzymane wzory, uzyskujemy związek

$$\begin{aligned} a_k &= \frac{W}{N} = \frac{W^+ + W^-}{N} = \frac{N^+ \frac{W^+}{N^+} + N^- \frac{W^-}{N^-}}{N} = \frac{N^+ a_{k+1} + N^- a_{k-1}}{N} = \\ &= pa_{k+1} + qa_{k-1}. \end{aligned}$$

Otrzymaliśmy więc bardzo prostą rekurencję liniową jednorodną rzędu 2:

$$\begin{aligned} a_k &= pa_{k+1} + qa_{k-1}, \\ a_0 &= 0, \quad a_n = 1, \quad (p + q = 1), \end{aligned} \tag{1.44}$$

lub pisząc w kolejności, do której przywykliśmy, $a_{k+1} = a_k/p - (q/p)a_{k-1}$. Rekurencję (1.44) możemy łatwo rozwiązać za pomocą Tw. 1.4. Równanie charakterystyczne ma postać

$$u = pu^2 + q,$$

zatem

$$u_1 = 1, \quad u_2 = \frac{q}{p}.$$

Założymy najpierw, że $p \neq q$ (w realistycznej sytuacji tak jest, bo aby prowadzenie kasyna się opłacało $p < q$). Przypadek $p < q$ nazywamy grą *niesprawiedliwą*. Wtedy na mocy Tw. 1.4 $a_k = c_1 u_1^n + c_2 u_2^n$. Pozostaje wyznaczyć stałe c_1 i c_2 , co czynimy za pomocą warunków początkowych (1.44):

$$\begin{aligned} a_0 &= 0 \Rightarrow c_1 = -c_2, \\ a_n &= 1 \Rightarrow c_1 = 1/(1 - (q/p)^n), \end{aligned}$$

więc ostatecznie

$$a_k = \frac{1 - \left(\frac{q}{p}\right)^k}{1 - \left(\frac{q}{p}\right)^n}, \quad (p \neq q). \tag{1.45}$$

Dla gry *sprawiedliwej*, $p = q = 1/2$, korzystamy z drugiego przypadku Tw. 1.4, który mówi, że $a_k = (c_1 + c_2 k)$. Warunki (1.44) dają teraz $c_1 = 0$, $c_2 = 1/n$, czyli

$$a_k = \frac{k}{n}, \quad (p = q = \frac{1}{2}). \tag{1.46}$$

Przypadek gry sprawiedliwej można też uzyskać w inny sposób jako granicę gry niesprawiedliwej. Istotnie, weźmy

$$p = \frac{1}{2} - \epsilon, \quad q = \frac{1}{2} + \epsilon$$

we wzorze (1.45), gdzie $\epsilon > 0$. Otrzymujemy wówczas

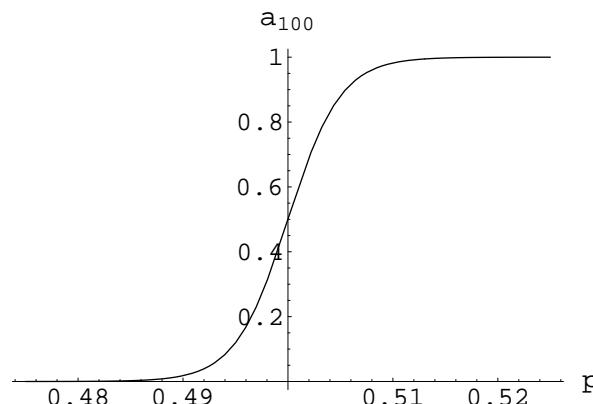
$$a_k = \frac{1 - \left(\frac{1+2\epsilon}{1-2\epsilon}\right)^k}{1 - \left(\frac{1+2\epsilon}{1-2\epsilon}\right)^n}.$$

W granicy $\epsilon \rightarrow 0$ jest to symbol nieokreślony $0/0$. Stosując regułę de L'Hospitala oraz fakt, że

$$\frac{d}{d\epsilon} \left(\frac{1-2\epsilon}{1+2\epsilon} \right)^m \Big|_{\epsilon=0} = -4m,$$

dstajemy ponownie $a_k = k/n$, w zgodzie z (1.46).

Teraz porozmawiamy o liczbach, które wynikają z powyższych wzorów. Jeśli $n = 200$ oraz $k = 100$, to dla gry sprawiedliwej $a_{100} = 1/2$, co wynika też natychmiast z symetrii zagadnienia. Natomiast jeśli $p = 0.49$, to zgodnie z wzorem (1.45) $a_k \simeq 2\%$, czyli szanse są bardzo małe. Tylko co pięćdziesiąty gracz wygrywa. Dla $p = 0.45$ już tylko dwóch graczy na miliard wygrywa! Widzimy więc, że szanse zwycięstwa są niezwykle czułe na odstępstwo p od wartości $1/2$, czyli na „niesprawiedliwość” korzystną dla kasyna. Efekt ten jest bardzo dobrze widoczny na rys. 1.10, gdzie pokazano zależność a_{100} od wartości p . Dla $p = q = 1/2$ (gra sprawiedliwa), $a_{100} = 1/2$. Odstępstwo w kierunku $p > 1/2$ powoduje bardzo szybkie dążenie $a_{100} \rightarrow 1$, czyli niemal pewność rozbicia banku. Odwrotnie, wartości $p < 1/2$ powodują również szybkie dążenie $a_{100} \rightarrow 0$, czyli rychłe „spłukanie” gracza!



Rysunek 1.10: Zależność prawdopodobieństwa rozbicia banku od wartości prawdopodobieństwa wygranej w pojedynczym rozdaniu, p . Widać bardzo czułe zachowanie ze względu na niewielkie odstępstwa p od wartości $1/2$ (rozbicie banku przy $n = 200$, początkowa ilość monet $k = 100$)

Większość ludzi uczęszczających kasyna nie po to, żeby wygrać – prawa statystyki są nieubłagane – ale by poprzebywać w miłym towarzystwie i przyjemnie spędzić czas. Powstaje więc pytanie: *jak długo będzie trwać ruina gracza?* Ścisłej, jaka jest wartość oczekiwana liczby rozdań od stanu, gdy gracz ma k monet, do stanu, gdy zostaje spłukany, lub też, co znacznie rzadsze, gdy rozbije bank. Oznaczmy tę wielkość jako t_k . Rozumowanie analogiczne do przeprowadzonego powyżej prowadzi do rekurencji

$$\begin{aligned} t_k &= 1 + pt_{k+1} + qt_{k-1}, \\ t_0 &= 0, \quad t_n = 0. \end{aligned} \tag{1.47}$$

Jest tu pewna istotna różnica w porównaniu z wzorem (1.44), a mianowicie po prawej stronie równania rekurencyjnego występuje dodatkowo jedynka (człon *niedojednorodny*). Jest to zrozumiałe, gdyż liczymy teraz liczbę rozdań. Gracz miał k monet, raz zagrał, stąd właśnie ta jedynka. Ponadto z prawdopodobieństwem p wygrał, ma $k+1$ monet i teraz oczekiwany czas dalszej gry to t_{k+1} , lub z prawdopodobieństwem q przegrał, ma $k-1$ monet i oczekiwany czas dalszej gry wynosi t_{k-1} . Warunki *początkowe* też są teraz inne niż w (1.44). Zgodnie z zasadami, gra się kończy, gdy gracz nie ma już monet lub gdy wygrał n monet. W obu tych przypadkach czas dalszej gry wynosi zero.

Pozostaje rozwiązać rekurencję (1.47). Skorzystamy tu z formalizmu przedstawionego w podrozdziale 1.6. Rozważmy przypadek gry niesprawiedliwej. Rozwiązanie *ogólne* stwarzyszonej rekurencji jednorodnej do (1.47) ma oczywiście tę samą postać, co dla równania (1.44), $j_k = c_1 u_1^k + c_2 u_2^k$, gdzie $u_1 = 1$, $u_2 = q/p$. Znajdźmy teraz *rozwiązanie szczegółowe*. Ponieważ człon niedojednorodny jest równy jednemu z pierwiastków równania charakterystycznego $u_1 = 1$, szukamy rozwiązania postaci $s_k = w_0 k$. Wstawienie do (1.47) daje

$$\begin{aligned} w_0 k &= 1 + pw_0(k+1) + qw_0(k-1), \\ 0 &= 1 + pw_0 - qw_0, \\ w_0 &= \frac{1}{q-p}. \end{aligned}$$

A zatem $t_k = c_1 u_1^k + c_2 u_2^k + k/(q-p)$. Warunki $t_0 = t_n = 0$ prowadzą do

$$0 = c_1 + c_2, \quad 0 = c_1 + c_2 \left(\frac{q}{p} \right)^n + \frac{n}{q-p},$$

skąd

$$c_1 = -c_2 = \frac{n}{(p-q) \left[1 - \left(\frac{q}{p} \right)^n \right]}$$

i ostatecznie

$$t_k = \frac{n \left[1 - \left(\frac{q}{p} \right)^k \right] - k \left[1 - \left(\frac{q}{p} \right)^n \right]}{(p-q) \left[1 - \left(\frac{q}{p} \right)^n \right]}. \tag{1.48}$$

Ponapawajmy się chwilę tym rozwiązaniem. Po pierwsze, możemy sprawdzić, że w istocie spełnia równanie (1.47). Podstawiając $k = 100$, $n = 200$ i $p = 0.49$ otrzymujemy średni oczekiwany czas gry. Jeśli zaczynamy ze stoma monetami, $k = 100$, to $t_{100} \simeq 4820$. Dla $p = 0.45$ czas ten topnieje do wartości $t_{100} \simeq 1000$. Widać tu, dlaczego gra nie może być zbyt niesprawiedliwa. Po prostu wtedy za krótko by trwała, co zniechęcałoby klientów kasyna.

Przypadek sprawiedliwy, $p = 1/2$, możemy uzyskać na dwa sposoby: albo zastosować formalizm dla przypadku, gdy pierwiastki równania charakterystycznego są sobie równe, albo też, co jest szybsze, zastosować podstawienie $p = 1/2 - \epsilon$, rozwiniąć w zmiennej ϵ i wziąć $\epsilon \rightarrow 0$. W wyniku dostajemy

$$t_k = k(n - k), \quad (p = q = \frac{1}{2}), \quad (1.49)$$

co dla $k = 100$ i $n = 200$ daje $t_k = 10000$.

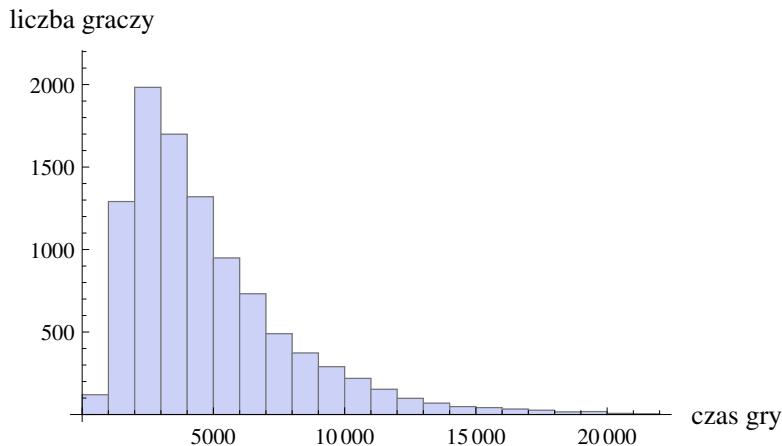
Ostatnie pytanie, które zadamy, to *jakie jest prawdopodobieństwo gry bez końca?* Oznaczmy to prawdopodobieństwo jako b_n . Powtarzając rozumowanie prowadzące do zanalizowanych powyżej problemów, dostajemy rekurencję

$$\begin{aligned} b_k &= pb_{k+1} + qb_{k-1}, \\ b_0 &= 0, \quad b_n = 0, \quad (p + q = 1). \end{aligned} \quad (1.50)$$

Warunki początkowe $b_0 = b_n = 0$ oznaczają tu tyle, że jeśli gra się skończyła, to prawdopodobieństwo jej trwania w nieskończoność wynosi 0. Wykonując analogiczną analizę jak w przypadku (1.44), otrzymujemy rozwiązanie $b_k = 0$ dla każdego k . Oznacza to, że gra zawsze się skończy, trzeba tylko dostatecznie długo grać. Matematycznie, warunki początkowe w (1.50) „*ubili*” rozwiązań. Zauważmy, że jedyna różnica między rekurencjami (1.44) i (1.50) tkwi właśnie w warunkach początkowych – widzimy namacalnie, jak bardzo są one istotne.

Uwaga 1.2. Przypominamy, że rekurencja to nie tylko przepis wyrażający n -ty wyraz poprzez wyrazy poprzednie, ale także warunki początkowe, które określają rozwiązańe.

Nasze spotkanie z hazardystami kończymy pokazaniem na rys. 1.11 symulacji numerycznej dla 10000 graczy, przedstawiającej rozkład czasu gry dla parametrów $p = 0.49$, $q = 0.51$, $k = 100$ i $n = 200$. Jest to *histogram*, który należy czytać w następujący sposób: podstawa słupka pokrywa zakres czasu gry, np. 1-1000 rozdań, 1001-2000 itd., natomiast wysokość słupka ukazuje liczbę graczy, dla których czas gry zmieścił się w tym zakresie. W szczególności ok. 1900 graczy grało od 2001 do 3000 rozdań, ok. 500 od 7001 do 8000 rozdań itd. Przy liczbie rozdań dążącej do nieskończoności widzimy długi „ogon” tego rozkładu: graczy, którzy grają dłużej, jest coraz mniej, jednak jeśli liczba graczy będzie zmierzać do nieskończoności, to zawsze znajdziemy gracza, który będzie grać dowolnie długo.



Rysunek 1.11: Histogram rozkładu czasu gry w kasynie dla symulacji numerycznej 10000 graczy ($p = 0.49$, $q = 0.51$, $k = 100$, $n = 200$)

1.8 Szybkie mnożenie i rekurencje „dziel i rządź”

Wspólną cechą dyskutowanych dotąd rekurencji było to, że „cofały” się w definicji n -tego wyrazu do poprzednich wyrazów, jednak nie dalej niż *stała liczba* wyrażająca różnicę wskaźników (tj. rząd rekurencji), np. ciąg Hanoi definiuje h_n za pomocą poprzedniego wyrazu h_{n-1} (cofa się o 1), ciąg Fibonacciego F_n za pomocą dwóch poprzednich wyrazów, F_{n-1}, F_{n-2} (cofa się o 2) itp. Istnieje ważna klasa rekurencji, w których to „cofanie” idzie znacznie głębiej, nie o kilka kroków, ale kilka *razy* w tył. W pewnych praktycznych sytuacjach możemy bowiem podzielić problem na pewną liczbę znacznie „mniejszych” problemów, które są łatwiejsze do rozwiązania. Określenie „mniejszy” dotyczy tu liczby n , która określa długość *danych wejściowych*. Zanim przejdziemy do skrupulatnej definicji i analizy rekurencji „dziel i rządź” (ang. *divide and conquer* – dziel i zwyciężaj!), rozważmy najpierw przykład doskonale nam znanego algorytmu mnożenia liczb w słupku, który nie jest algorytmem tego typu.

Przykład 1.1 (mnożenie w słupku). *Musimy pomnożyć przez siebie „na piechotę” dwie wielocyfrowe liczby, bo nie mamy pod ręką kalkulatora albo też wynik jest zbyt długi i nie mieści się na jego wyświetlaczu! Weźmy np.*

$$w = 88438928 \cdot 81287298.$$

Standardowy algorytm „mnożenia w słupku”, którego uczymy się w szkole, działa następująco:

$$\begin{array}{r}
 88438928 \\
 \cdot 81287298 \\
 \hline
 707511424 \\
 795950352 \\
 176877856 \\
 619072496 \\
 707511424 \\
 176877856 \\
 88438928 \\
 +707511424 \\
 \hline
 = 7188961495136544
 \end{array}$$

Oznaczmy długość (liczbę cyfr) liczb jako $n = 8$. W powyższym obliczeniu w każdym pośrednim wierszu (wiersz pośredni to wiersz między dwiema kreskami) wykonaliśmy n mnożeń liczb jednocyfrowych oraz maksymalnie $n - 1$ dodawań liczb jednocyfrowych, czynionych gdy wynik mnożenia jest większy od 10. Na przykład $88438928 \cdot 8$ liczymy jako $8 \cdot 8 = 64$, 4 zapisujemy jako ostatnią cyfrę, 6 zapamiętujemy na chwilę i dodajemy do wyniku kolejnego mnożenia $8 \cdot 2 = 16$ itd. Pamiętamy to doskonale z lekcji arytmetyki! Ponieważ wierszy pośrednich jest n , łączny czas ich obliczania jest rzędu $\Theta(n^2)$ (zob. def. 1.6). Na koniec musimy jeszcze dodać do siebie n wierszy pośrednich, co wymaga rzędu n^2 dodawań liczb jednocyfrowych, zatem całkowity czas wykonywania algorytmu jest rzędu $\Theta(n^2)$.

Jest to więc algorytm o czasie wykonywania rosnącym kwadratowo z długością mnożonych liczb. Okazuje się jednak, że istnieją znacznie efektywniejsze algorytmy mnożenia długich liczb. Przykładem jest opublikowany w 1962 r. [38] algorytm szybkiego mnożenia Karatsuby¹³ i Ofmana¹⁴, który teraz omówimy.

Przykład 1.2 (szybkie mnożenie Karatsuby i Ofmana). Zapiszmy mnożenie z poprzedniego przykładu jako

$$\begin{aligned}
 w &= 88438928 \cdot 81287298 = (10^4 8843 + 8928) \cdot (10^4 8128 + 7298) \\
 &= 10^8 8843 \cdot 8128 + 10^4 (8843 \cdot 7298 + 8928 \cdot 8128) + 8928 \cdot 7298.
 \end{aligned}$$

Powyższa równość wygląda nieco przejrzystiej, jeśli wprowadzimy oznaczenia $x_1 = 8843$, $x_2 = 8928$, $y_1 = 8128$, $y_2 = 7298$:

$$\begin{aligned}
 w &= (10^4 x_1 + x_2) \cdot (10^4 y_1 + y_2) \\
 &= 10^8 x_1 y_1 + 10^4 (x_1 y_2 + x_2 y_1) + x_2 y_2.
 \end{aligned} \tag{1.51}$$

„Przełamaliśmy” nasze liczby na pół, co samo w sobie nic jeszcze nie daje. Ale teraz kluczowa i genialna w swej prostocie obserwacja Karatsuby i Ofmana: sumę

¹³ Anatoli Aleksiejewicz Karatsuba (1937-), rosyjski matematyk.

¹⁴ Yuri P. Ofman (1939-), rosyjski matematyk.

iloczynów w środkowym członie w (1.51) można zapisać jako

$$x_1y_2 + x_2y_1 = (x_1 + x_2)(y_1 + y_2) - x_1y_1 - x_2y_2. \quad (1.52)$$

Tak więc w celu znalezienia wyniku w potrzebujemy obliczyć trzy iloczyny: $A = x_1y_1$, $B = x_2y_2$, $C = (x_1 + x_2)(y_1 + y_2)$, jedną różnicę $D = C - A - B$, a także sumę $10^8A + 10^4D + B$. Problem zredukowaliśmy zatem do policzenia dwóch iloczynów liczb czterocyfrowych, A i B , jednego iloczynu liczb co najwyżej pięciocyfrowych, C , policzenia różnicy D oraz wykonania sumy w (1.51). Iloczyn liczb pięciocyfrowych to w naszym przypadku $16971 \cdot 16226$. Możemy go rozbić w następujący sposób:

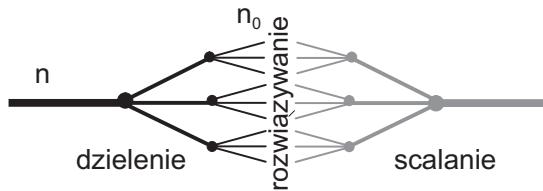
$$(1 \cdot 10^5 + 6971)(1 \cdot 10^5 + 6226) = 1 \cdot 1 \cdot 10^{10} + 1 \cdot (6971 + 6226)10^5 + 6971 \cdot 6226.$$

Mamy tu ponownie mnożenie dwóch liczb czterocyfrowych oraz dodatkowe operacje: iloczyn liczb jednocyfrowych, sumę liczb czterocyfrowych i ich mnożenie przez liczbę jednocyfrową, wreszcie sumę wszystkich trzech przyczynków. Te dodatkowe operacje zajmują czas proporcjonalny do liczby cyfr, czyli są $\mathcal{O}(n)$. Na koniec musimy jeszcze wykonać sumę we wzorze (1.51), co również jest $\mathcal{O}(n)$. Ostatecznie początkowy problem mnożenia liczb ośmiocyfrowych został zredukowany do (uwaga!) trzech problemów mnożenia liczb czterocyfrowych plus pewna dodatkowa praca proporcjonalna do liczby cyfr. Oznaczając czas potrzebny do pomnożenia dwóch liczb n -cyfrowych jako $T(n)$, możemy w ogólności napisać związek

$$T(n) = 3T(n/2) + \alpha n, \quad \alpha > 0. \quad (1.53)$$

Konkretną wartość współczynnika α można znaleźć, ale nie jest ona istotna dla dalszych rozważań. Podzieliliśmy więc nasz problem o $n = 8$ na trzy problemy o $n/2 = 4$. Rzeczą jasna, możemy działać rekurencyjnie i dzielić dalej na problemy o $n/4 = 2$, wreszcie $n/8 = 1$, gdzie dochodzimy do elementarnego mnożenia dwóch liczb jednocyfrowych. Za chwilę zrozumiemy, dlaczego (dla dostatecznie dużych liczb) zyskujemy na czasie wykonując algorytm Karatsuby i Ofmana w porównaniu z mnożeniem w słupku.

Przykład 1.2 obrazuje ideę algorytmów „dziel i rządź”: początkowy problem dzielony jest rekurencyjnie na mniejsze podproblemy. Po dojściu do pewnego minimalnego $n = n_0$ problemy są rozwiązywane, po czym wyniki są scalane, aby w efekcie dać rozwiązanie problemu wyjściowego. W przykładzie 1.2 każdy problem dzielony jest na $a = 3$ podproblemy, każdy o $b = 2$ razy krótszych danych wejściowych. Czas scalania jest proporcjonalny do n . Sytuacja przedstawiona jest na rys. 1.12.



Rysunek 1.12: Algorytm „dziel i rządź” – szybkie mnożenie Karatsuby i Ofmana. Początkowy problem o długości danych n dzielony jest na trzy podproblemy o długości $n/2$, te są dzielone na kolejne „krótsze” podproblemy itd., aż do momentu, gdy długość problemu wynosi n_0 . Wówczas problemy są rozwiązywane, a następnie ich wyniki są scalane, dając w efekcie rozwiązanie wyjściowego problemu

A teraz bardziej formalna definicja:

Def. 1.14 (rekurencja „dziel i rządź”). Najprostsza rekurencja „dziel i rządź” ma postać

$$\begin{aligned} T(n) &= aT\left(\frac{n}{b}\right) + g(n), \quad a \geq 1, b = 1, 2, 3, \dots \\ T(n_0) &= \Theta(1), \quad n_0 \geq 1, \end{aligned}$$

gdzie a jest liczbą podproblemów, na które dzielimy wyjściowy problem, b określa ile razy podproblem jest „mniejszy” od problemu początkowego, natomiast $g(n)$ jest pewną funkcją określającą liczbę dodatkowych operacji niezbędnych do podzielenia i scalenia problemu.

Oszacujemy teraz czas wykonywania rekurencji 1.14. Problem ten rozstrzyga tzw. twierdzenie o rekurencji uniwersalnej [4], którego uproszczony wariant jest następujący:

Tw. 1.8 (o rekurencji uniwersalnej, uproszczone). Rozważmy $g(n) = \alpha n^p$, $\alpha > 0$. Wówczas rozwiązanie rekurencji 1.14 ma następujące zachowanie asymptotyczne:

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{dla } p < \log_b a \\ \Theta(n^{\log_b a} \log_b n) & \text{dla } p = \log_b a \\ \Theta(n^p) & \text{dla } p > \log_b a \end{cases}. \quad (1.54)$$

Dowód: Bez utraty ogólności możemy przyjąć $n_0 = 1$. W powyższym sformułowaniu twierdzenia argument funkcji T musi być liczbą całkowitą. Tkwi więc tutaj ukryte założenie: n musi być postaci

$$n = b^i, \quad i = 0, 1, 2, \dots$$

Dzięki temu w kolejnych krokach rekurencyjnych zawsze jesteśmy w stanie podzielić liczbę określającą długość danych przez b , dostając kolejno $b^i, b^{i-1}, \dots, 1$.

Oznaczmy teraz

$$r_i = T(n) = T(b^i).$$

Wówczas rekurencja 1.14 przyjmuje postać

$$r_i = ar_{i-1} + \alpha b^{ip}.$$

Otrzymaliśmy więc dobrze nam znaną niejednorodną rekurencję liniową z Tw. 1.6-1.7! Równanie charakterystyczne jest postaci $u = a$, ma więc jeden pierwiastek a , natomiast stała t z równania (1.41) wynosi $t = b^p$, ponadto $m = 0$ oraz $k_{\max} = 1$. W związku z tym asymptotyka określona jest przez największą liczbę spośród zbioru $\{a, b^p\}$. Na mocy wniosku 1.3, jeśli $a > b^p$, to $r_i \sim a^i$. Warunek możemy zapisać równoważnie jako $p < \log_b a$, a postać asymptotyczną w formie $T(n) = r_i \sim a^i = a^{\log_b n} = n^{\log_b a}$, w czym rozpoznajemy pierwszy przypadek równania (1.54). Dla $a = b^p$ wniosek 1.3 prowadzi do $T(n) = r_i \sim i a^i = \log_b n n^{\log_b a}$, co stanowi drugi przypadek tezy. Wreszcie dla $a < b^p$ znajdujemy, że $T(n) = r_i \sim t^i = b^{pi} = n^p$, co jest ostatnim przypadkiem równania (1.54). \square

Możemy teraz powrócić do szybkiego algorytmu mnożenia. Rekurencja (1.53) przyjmuje $a = 3$, $b = 2$ oraz $p = 1$. Ponieważ $\log_2 a > p$, mamy pierwszy przypadek z Tw. 1.8, zatem asymptotycznie

$$T(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.585\dots}).$$

Jest to dla dużych n zdecydowanie wolniejszy wzrost niż dla mnożenia w słupku, rosnącego jak n^2 . Zauważmy, że spostrzeżenie (1.52) redukujące liczbę mnożeń z czterech do trzech jest absolutnie kluczowe. Gdybyśmy wykonywali cztery mnożenia, to obowiązywałaby rekurencja $T(n) = 4T(n/2) + \alpha n$. Wówczas $T(n) \sim n^{\log_2 4} = n^2$, tak samo jak dla mnożenia w słupku – nic byśmy nie zyskali!

Powstaje więc pytanie, dlaczego uczymy się w szkole zwykłego mnożenia w słupku, a nie szybkiego mnożenia 1.2. Otóż uzyskane wyżej oszacowania są słuszne *asymptotycznie*. Dla małych wartości n istotną rolę odgrywają operacje potrzebne na podzielenie i scalenie problemu, dzięki czemu zwykłe mnożenie „wygrywa” z algorymem Karatsuby i Ofmana dla mnożników poniżej ok. 10^{96} . Należy wspomnieć, że są jeszcze szersze algorytmy mnożenia. Na przykład algorytm Schönhage'a¹⁵ i Strassena¹⁶, oparty na tzw. szybkich transformatach fourierowskich, działa w czasie $\mathcal{O}(n \log n \log(\log n))$.

Dowód Tw. 1.8 pokazał, że w zasadzie nie ma różnic między przytoczonym wariantem twierdzenia a zwykłą niejednorodną rekurencją liniową rzędu jeden. Było tak dlatego, że rozważaliśmy szczególny przypadek $n = b^i$. Twierdzenie

¹⁵ Arnold Schönhage (1934-), niemiecki matematyk i informatyk.

¹⁶ Volker Strassen, (1936-), matematyk niemiecki.

można jednak uogólnić. Zanim to zrobimy, zdefiniujemy kilka użytkowych funkcji całkowitoliczbowych, tzn. funkcji odwzorowujących zbiór liczb rzeczywistych w zbiór liczb całkowitych.

Def. 1.15. Funkcja „podłoga” (ang. floor), oznaczona jako $\lfloor x \rfloor$, to największa całkowita mniejsza lub równa x . Funkcja „sufit” (ang. ceiling), $\lceil x \rceil$, to najmniejsza całkowita większa lub równa x .

Jest oczywiste, że dla liczby całkowitej x zachodzi $\lfloor x \rfloor = \lceil x \rceil$, a dla liczby x niebędącej liczbą całkowitą, $\lfloor x \rfloor = \lceil x \rceil - 1$.

Def. 1.16. Funkcja „część całkowita” (ang. integer) to

$$\lfloor x \rfloor = \begin{cases} \lfloor x \rfloor & \text{dla } x \geq 0 \\ \lceil x \rceil & \text{dla } x < 0 \end{cases}.$$

Wartość tej funkcji otrzymujemy z liczby poprzez odrzucenie jej *mantisy*, czyli cyfr po kropce dziesiętnej¹⁷.

Def. 1.17. Funkcja „najbliższa liczba całkowita”, służąca zaokrąglaniu,

$$\lceil x \rceil$$

jest zdefiniowana jako liczba całkowita najbliższa liczbie x , przy czym dla liczb połówkowych zaokrąglamy do najbliższej liczby parzystej, tj.

$$\begin{aligned} \lceil n + 1/2 \rceil &= n + 1, & (n \text{ nieparzyste}), \\ \lceil n + 1/2 \rceil &= n, & (n \text{ parzyste}). \end{aligned}$$

Funkcje całkowitoliczbowe są często używane w teorii liczb, umożliwiając zwarty zapis twierdzeń dotyczących liczb całkowitych.

A teraz uogólnienie Tw.1.8, które m.in. uwalnia nas od założenia, że $n = b^i$:

Tw. 1.9 (o rekurencji uniwersalnej). Rozważmy dowolną asymptotycznie dodatnią funkcję $g(n)$ oraz $a \geq 1$ i $b > 1$. Uwaga: niech n/b oznacza podłogę $\lfloor \frac{n}{b} \rfloor$ lub sufit $\lceil \frac{n}{b} \rceil$. Wówczas rozwiązanie rekurencji

$$T(n) = aT(n/b) + g(n)$$

ma następujące zachowanie asymptotyczne:

$$T(n) = \begin{cases} 1. \Theta(n^{\log_b a}) & \text{gdy } \exists \epsilon > 0 : g(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \\ 2. \Theta(n^{\log_b a} \log^{k+1} n) & \text{gdy } g(n) = \Theta(n^{\log_b a} \log_b^k n), k \geq 0, \\ 3. \Theta(n^p) & \text{gdy } \exists \epsilon > 0 : g(n) = \Omega(n^{\log_b a + \epsilon}) \\ & \text{oraz } \exists c < 1 \exists n_0 \forall n > n_0 : cg(n) \geq ag(n/b). \end{cases}$$

¹⁷ Kropka dziesiętna wyparła w notacji naukowej przecinek dziesiętny!

Dowód można znaleźć w [4]. Warunek w przypadku 1 oznacza, że $g(n)$ rośnie wolniej niż $n^{\log_b a}$. W przypadku 2 asymptotyczny wzrost $g(n)$ musi być taki, jak $n^{\log_b a}$ pomnożone przez nieujemną potęgę $\log_b n$. W trzecim przypadku $g(n)$ rośnie szybciej niż $n^{\log_b a}$, a dodatkowy warunek, tzw. warunek regularności, oznacza, że dla dostatecznie dużych n możemy dobrać taką stałą c , że zachodzi $cg(n) \geq ag(n/b)$. Warunek ten zachodzi dla większości funkcji spełniających pierwszy warunek $g(n) = \Omega(n^{\log_b a + \epsilon})$, tj. dla wielomianów, funkcji wykładniczej, $n!$, ich iloczynów itp.

Zauważmy, że funkcja $g(n) = n^{\log_b a} / \log(n)$ „wpada” pomiędzy przypadek 1 a 2, a funkcja $g(n) = n^{\log_b a} \log \log n$ pomiędzy przypadki 2 a 3. Twierdzenie nie pokrywa więc wszystkich możliwych wyborów funkcji $g(n)$, jednak stosuje się do większości sytuacji „z życia”. Dzięki temu Tw. 1.9 jest najpopularniejszym narzędziem analizy złożoności algorytmów typu „dziel i rządź”. Porównując Tw. 1.9 i 1.8 zauważamy też, że niewielkie zmiany w rekurencji (np. wprowadzenie funkcji podloga i sufit zamiast zwykłego dzielenia) nie wpływają na zachowanie asymptotyczne rozwiązania.

Na koniec tego podrozdziału przedstawimy bardziej zaawansowany materiał, opisujący jeszcze ogólniejszą metodę analizy rekurencji „dziel i rządź”, stosowaną do bardzo szerokiej klasy przypadków. Eleganckiego aparatu dostarcza otrzymane zupełnie niedawno (1996 r.) Tw. Akry¹⁸-Bazziego¹⁹, stosowane również do rekurencji typu „dziel i rządź”, gdzie rozmiar podproblemów nie musi być jednakowy. Przytaczamy tu wersję tego twierdzenia z pracy [39].

Tw. 1.10 (Akry-Bazziego). *Rozważmy rekurencję postaci*

$$T(x) = \begin{cases} \Theta(1) & \text{dla } 1 \leq x \leq x_0 \\ \sum_{i=1}^k a_i T(\beta_i x) + g(x) & \text{dla } x > x_0 \end{cases}, \quad (1.55)$$

gdzie $x \geq 1$ jest zmienną rzeczywistą, a $x_0, a_i > 0$ oraz $0 < \beta_i < 1$ ($i = 1, \dots, k$) są stałymi rzeczywistymi. Ponadto zachodzą następujące warunki:

- $x_0 \geq 1/\beta_i$ oraz $x_0 \geq 1/(1 - \beta_i)$, $i = 1, 2, \dots, k$,
- $g(x) \geq 0$,
- $\exists c_1, c_2 > 0 \forall x > x_0 \forall u \in (\beta_i x, x) : c_1 g(x) \leq g(u) \leq c_2 g(x)$ – jest to tzw. warunek wielomianowości.

Niech liczba p będzie rozwiązaniem równania

$$\sum_{i=1}^k a_i \beta_i^p = 1. \quad (1.56)$$

Wtedy

$$T(x) = \Theta\left(x^p \left[1 + \int_{x_0}^x \frac{g(u)}{u^{p+1}} du\right]\right). \quad (1.57)$$

¹⁸ Mohamad A. Akra, libański informatyk.

¹⁹ Louay Bazzi, libański informatyk.

Dowód można znaleźć w [39]. Podajmy kilka komentarzy i przykładów. Używamy teraz liczb rzeczywistych jako argumentów funkcji T , co jest uogólnieniem w stosunku do poprzednich twierdzeń. Założenie wielomianowości spełnione jest m.in. dla funkcji postaci $g(x) = x^\alpha \log^\gamma x$, gdzie α i γ są dowolnymi liczbami rzeczywistymi. Oprócz sprawdzenia założeń, musimy znaleźć liczbę p z warunku (1.56). Liczba ta jest jednoznacznie wyznaczona, ponieważ z założeń odnośnie β_i i a_i wynika, że $s(p) = \sum_{i=1}^k a_i \beta_i^p$ jest funkcją silnie malejącą zmiennej p oraz $\lim_{p \rightarrow -\infty} s(p) = \infty$ i $\lim_{p \rightarrow \infty} s(p) = 0$. W związku z tym $s(p) = 1$ w dokładnie jednym punkcie. Liczba p „kontroluje” asymptotykę razem z postacią funkcji g poprzez dość skomplikowany związek (1.57), zawierający całkę. Twierdzenie 1.10 dla $k = 1$ obejmuje jako szczególny przypadek Tw. 1.8 dla wielomianowych funkcji $g(x)$.

A teraz przykłady:

Przykład 1.3. Rozwiążmy

$$T(x) = 2T\left(\frac{x}{2}\right) + 3T\left(\frac{x}{3}\right) + 6T\left(\frac{x}{6}\right) + x^2 \log x.$$

Warunek (1.56) przyjmuje postać

$$2\left(\frac{1}{2}\right)^p + 3\left(\frac{1}{3}\right)^p + 6\left(\frac{1}{6}\right)^p = 1,$$

dający rozwiązanie²⁰ $p = 2$. Następnie obliczamy całkę

$$\int_1^x \frac{g(u)}{u^{p+1}} du = \int_{x_0}^x \frac{\log u}{u} du = \frac{1}{2} \log^2 u \Big|_{u=x_0}^x = \frac{1}{2} (\log^2 x - \log^2 x_0).$$

Na mocy twierdzenia znajdujemy $T(x) = \Theta(x^2 \log^2 x)$.

Przykład 1.4. Inny przykład to rekurencja

$$T(x) = 2T\left(\frac{x}{2}\right) + x/\log x,$$

dla której $p = 1$, oraz

$$T(x) = \Theta\left(x \left[1 + \int_{x_0}^x \frac{1}{u \log u} du\right]\right) = \Theta(x \log(\log x)).$$

Zauważmy pojawienie się podwójnego logarytmu.

Podobnie jak w Tw. o rekurencji uniwersalnej, niewielkie zmiany w argumentie funkcji T po prawej stronie związku rekurencyjnego nie wpływają na zachowanie asymptotyczne, które określa uogólnione Tw. Akry-Bazziego (zob. [39]).

²⁰ Dla dowolnie wybranych a_i i β_i analityczne rozwiązanie równania może być niemożliwe, ale stałą p można z dowolną dokładnością znaleźć numerycznie.

Tw. 1.11 (Akry-Bazziego, uogólnione). *Rozważmy rekurencję*

$$T(x) = \begin{cases} \Theta(1) & \text{dla } 1 \leq x \leq x_0 \\ \sum_{i=1}^k a_i T(\beta_i x + h_i(x)) + g(x) & \text{dla } x > x_0 \end{cases}, \quad (1.58)$$

gdzie spełnione są warunki Tw. 1.10, warunek wielomianowości $g(x)$ zachodzi na przedziale $u \in [\beta_i x + h_i(x), x]$, $i = 1, \dots, k$, ponadto funkcje $h_i(x)$ są ograniczone w następujący sposób:

$$\exists \epsilon > 0 \ \forall x > x_0 : |h(x)| < \frac{x}{\log^{1+\epsilon} x}.$$

Dodatkowo, przyjmuje się pewne techniczne założenia dotyczące wyboru x_0 (zob. [39]). Wtedy zachodzi ta sama teza, co w oryginalnym Tw. 1.10.

W szczególności co ma praktyczne znaczenie, założenia powyższego twierdzenia spełnione są dla $h_i(x) = x - \lceil x \rceil$ oraz $h_i(x) = x - \lfloor x \rfloor$.

Na tym kończymy nasze rozważania o rekurencji, do której jednak będziemy często powracać w dalszych częściach wykładu, albowiem wiele problemów matematyki dyskretnej sprowadza się do znanych zagadnień rekurencyjnych. Dalsze informacje dotyczące tej tematyki można znaleźć w podręcznikach [2–4].

Ćwiczenia

- 1.1. Rozwiąż rekurencję $a_n = a_{n-1} + 2a_{n-2}$ dla $n > 2$ z warunkami początkowymi $a_1 = a_2 = 1$ za pomocą funkcji tworzącej oraz metody opartej na równaniu charakterystycznym.
- 1.2. Podczas rozwiązywania zadań poczuliśmy silny głód i zamówiliśmy pizzę. Jaka jest największa liczba kawałków p_n , na które możemy podzielić pizzę z pomocą n prostoliniowych cięć nożem? Inaczej: na jaką największą liczbę obszarów możemy podzielić płaszczyznę za pomocą n prostych? [3]
- 1.3. Liczby Lucasa: rozważ ciąg $L_n = F_n + 2F_{n-1}$, gdzie F_n są liczbami Fibonacciego, a L_n tzw. liczbami Lucasa. Znajdź rekurencję wyrażającą L_n poprzez L_{n-1} i L_{n-2} oraz stosowne warunki początkowe, a następnie rozwiąż tę rekurencję.
- 1.4. Pokaż, że liczby Fibonacciego i liczby Lucasa spełniają związek $F_{2n} = F_n L_n$.

1.5. Udowodnij następujące związki dla liczb Fibonacciego:

- (a) $\sum_{i=0}^n F_i = F_0 + F_1 + F_2 + \cdots + F_n = F_{n+2} - 1,$
- (b) $\sum_{i=0}^{n-1} F_{2i+1} = F_1 + F_3 + F_5 + \cdots + F_{2n-1} = F_{2n},$
- (c) $\sum_{i=1}^n i F_i = F_1 + 2F_2 + 3F_3 + \cdots + nF_n = nF_{n+2} - F_{n+3} + 2,$
- (d)
$$\begin{aligned} \sum_{i=1}^n i F_{2(n-i)+1} &= \\ &= F_{2n-1} + 2F_{2n-3} + 3F_{2n-5} + \cdots + (n-2)F_3 + (n-1)F_1 = F_{2n+1} - 1, \end{aligned}$$
- (e) $\sum_{i=0}^n F_i^2 = F_0^2 + F_1^2 + F_2^2 + \cdots + F_n^2 = F_n F_{n+1},$

gdzie $F_0 = 0.$

1.6. Pokaż, że dla $n \geq 0$ oraz $0 \leq i \leq n-1$ zachodzi związek

$$F_n = F_{i+1} F_{n-i} + F_i F_{n-i-1}.$$

1.7. Oblicz ułamek łańcuchowy

$$1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \dots}}}}$$

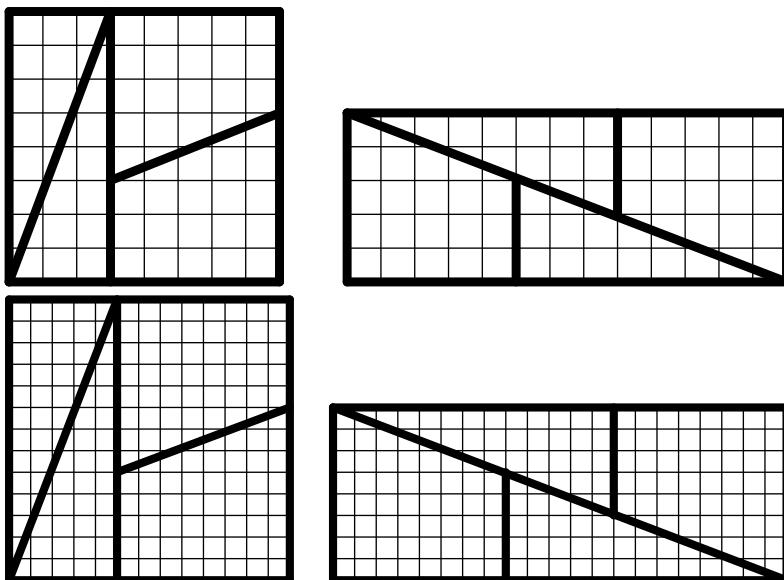
1.8. Używając dowolnej metody, rozwiąż rekurencję trzeciego stopnia

$$a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$$

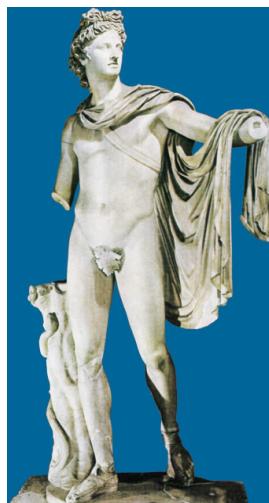
z warunkami początkowymi $a_0 = 0, a_1 = a_2 = 1.$

- 1.9. Rozwiąż rekurencję niejednorodną $a_n = a_{n-1} + 2a_{n-2} + (-1)^n$ dla $n > 2$ z warunkami początkowymi $a_1 = a_2 = 1$
- 1.10. Załóż, że masa kolejnych krążków Wież Hanoi rośnie jak n^2 – jest to dość realistyczne, bo typowa zabawka ma krążki o promieniach z grubsza wzrastających jak n i o tej samej grubości. Jaką całkowitą masę trzeba przenieść przy wykonaniu algorytmu 1.1?
- 1.11. Jaka jest największa liczba obszarów powstająca przy podziale trójwymiarowej przestrzeni n płaszczyznami?
- 1.12. Rozwiąż rekurencję $a_{n+1} = a_n^2$, gdzie $n \geq 1$, z warunkiem $a_1 = 2.$

- 1.13. Na okręgu wybrano (losowo) n różnych punktów, a między nimi narysowano wszystkie możliwe cięciwy. Na ile obszarów zostało podzielone koło?
- 1.14. Wiedząc, że truteń płodzony jest bezpłciowo z królowej, a królowa płodzona jest płciowo z trutnia i królowej, narysuj drzewo genealogiczne przodków trutnia. Policz liczbę osobników w każdym pokoleniu tego drzewa. Co odkryłeś? Udowodnij odkryty fakt.
- 1.15. Zmodyfikuj model rozmnażania królików Fibonacciego tak, aby każdy miot zawierał k par. Zapisz i rozwiąż stosowną rekurencję.
- 1.16. Zmodyfikuj model rozmnażania królików Fibonacciego tak, aby młode dojrzewały m razy dłużej niż czas trwania ciąży. Zapisz stosowną rekurencję i równanie charakterystyczne.
- 1.17. Korzystając z (1.9) możemy z łatwością sprawdzić, że np. $5 \cdot 13 - 8^2 = 1$, $8 \cdot 21 - 13^2 = -1$ itd. Twierdzenie 1.3 jest podstawą zabawnej łamigłówki, przedstawionej na rys. 1.13. Licząc kwadraciki na lewej części górnego rysunku, otrzymujemy $8 \cdot 8 = 64$, podczas gdy zliczenie po prawej stronie daje $5 \cdot 13 = 65$. Dzięki rozcięciu na kawałki zyskaliśmy jedną kratkę! Natomiast dla dolnego przypadku po lewej stronie $13 \cdot 13 = 169$, a po prawej $8 \cdot 21 = 168$. Tym razem gubimy jedna kratkę! Gdzie leży oszustwo?
- 1.18. Nostalgiczna powtórka z geometrii: udowodnij, że przekątne pentagramu dzielą się w stosunku, będącym złotym podziałem, zob. rys. 1.7.



Rysunek 1.13: Dwie łamigłówki oparte o tożsamość Cassiniego, pokazujące paradoksalnie, że $64 = 65$ i $169 = 168$! Kwadraty po lewej stronie rozcinamy wzduż linii i składamy tak, jak ukazano po prawej stronie. Dla górnego rys. 1.13 zliczenie kratek po lewej stronie daje $8 \cdot 8 = 64$, podczas gdy po prawej stronie mamy $5 \cdot 13 = 65$. Podobnie, dla dolnego przypadku z rys. 1.13 otrzymujemy $13 \cdot 13 = 169$, a $8 \cdot 21 = 168$



Rysunek 1.14: Apollo Belwederski (<http://hamlet.edu.pl/shi/galeria/show.php?id0=3&id1=0>). Stosunek R wzrostu do odległości pępką od podstawy jest bliski ϕ . Jeśli zmierzyć na rysunku wzrost od czubka prawej stopy do połowy czupryny, dostajemy $R = 1.61 \simeq \phi$ z dokładnością pomiaru ok. 1%

- 1.19. Z linijką w ręce doszukaj się złotego podziału w otaczających Cię przedmiotach.
- 1.20. Apollo Belwederski (zob. rys. 1.14 Leocharesa²¹, zob. rys. 1.14, ucieleśnia ideał proporcji ciała ludzkiego. W szczególności stosunek jego wzrostu do odległości pępką od podstawy jest bardzo bliski złotemu podziałowi ϕ . Dokonaj tego pomiaru na sobie!
- 1.21. Gracz wchodzi do kasyna, posiadając 100 monet, rozbicie banku następuje, gdy gracz posiądzie 200 monet. Prawdopodobieństwo wygranej wynosi $p = 0.49$. Krzysztof ma strategię taką, że wychodzi po wygraniu 50 monet, a Ludwik po przegraniu 50 monet. Jakie jest prawdopodobieństwo tych zdarzeń? Który z graczy będzie grał dłużej?
- 1.22. Jakie jest prawdopodobieństwo, że gracz w zadaniu o ruinie gracza osiągnie w trakcie gry ponownie swój początkowy stan posiadania?
- 1.23. Pewien profesor z Krakowa po dwóch piwach przechadza się ul. Sienkiewicza w Kielcach i na każdym skrzyżowaniu zupełnie losowo idzie albo w kierunku dworca PKP, albo w kierunku przeciwnym, gdzie ul. Sienkiewicza rozciąga się w nieskończoność. Jakie jest prawdopodobieństwo i średni czas dotarcia na dworzec?
- 1.24. Rozważ uogólnienie algorytmu Karatsuby i Ofmana na przypadek, gdzie liczba n dzielona jest na trzy mniejsze liczby [16] (tzw. algorytm Tooma²²

²¹ Leochares, IV w. pne., grecki rzeźbiarz.

²² Andrej Toom (1942-), rosyjski matematyk.

i Cooka²³). Jaka jest złożoność tego algorytmu, czyli asymptotyczna zależność czasu wykonywania od n dla dużych n ? Jakie jest uogólnienie, gdy podział dokonywany jest na s liczb?

- 1.25. Znajdź zachowanie asymptotyczne rekurencji

$$T(x) = \frac{1}{2}T\left(\frac{x}{2}\right) + 2T\left(\frac{x}{4}\right) + x^\alpha.$$

- 1.26. Rozważ algorytm *przeszukiwania binarnego* (ang. *binary search*) uporządkowanej rosnąco listy L o długości n , zawierającej obiekty p_i . Algorytm działa w następujący sposób: szukamy obiektu a , który jest zawarty gdzieś w liście. Porównujemy go z obiektem $b = p_{\lfloor n/2 \rfloor}$, tj. obiektem najbliższej średka listy. Jeśli $a = b$, kończymy, bo znaleźliśmy nasz obiekt. Jeśli $a > b$, przeszukujemy dolną część listy, tj. poniżej b . Jeśli $a < b$, przeszukujemy górną część listy. Czynność powtarzamy rekurencyjnie. Znajdź złożoność algorytmu.
- 1.27. Rozważ algorytm *sortowania przez scalanie* (ang. *merge sort*) listy L o długości n , w którym lista dzielona jest rekurencyjnie na dwie mniejsze listy L_1 i L_2 o długościach $\lceil n/2 \rceil$ oraz $\lfloor n/2 \rfloor$, każda z tych list jest sortowana, a wynik jest scalany w następujący sposób: 1) tworzymy wskaźniki w_1 i w_2 oraz ustawiamy je na początku list L_1 i L_2 , $w_1 = 1$, $w_2 = 1$. Tworzymy pustą listę wynikową K . 2) Jeżeli $L_1(w_1) \leq L_2(w_2)$, dopisujemy $L_1(w_1)$ na końcu K i zwiększamy w_1 o jeden, w przeciwnym przypadku dopisujemy $L_2(w_2)$ na końcu K i zwiększamy w_2 o jeden. Powtarzamy krok 2) aż wszystkie wyrazy z list L_1 i L_2 zostaną wpisane do K . Zapisz stosowną rekurencję „dziel i rządź” i oszacuj jej czas wykonywania dla dużych n . Algorytm ten pochodzi od Johna von Neumanna²⁴.

²³ Stephen Arthur Cook (1939-), amerykański informatyk.

²⁴ John von Neumann (1903-1957), urodzony na Węgrzech i pracujący w USA matematyk i fizyk. Przyczynił się bardzo istotnie do rozwoju analizy funkcjonalnej, mechaniki kwantowej, teorii mnogości, topologii, teorii gier, informatyki, statystyki, a także hydrodynamiki reakcji termojądrowych, które badał w ramach projektu Manhattan.

Rozdział 2

Kombinatoryka

Kombinatoryka jest działem matematyki zajmującym się zliczaniem elementów zbiorów skończonych. Ma bezpośrednie zastosowanie w bardzo wielu dziedzinach matematyki: w rachunku prawdopodobieństwa, algebrze, topologii, geometrii, teorii grafów, analizie algorytmów, a także w fizyce statystycznej, genetyce czy lingwistyce teoretycznej. Słowo kombinatoryka pochodzi od kombinowania, czyli łączenia, konstruowania. Rzecz w tym, że zbiory, których elementy będziemy zliczać, mogą być określone w bardzo zawiły, niejawnym sposobie, wobec czego zliczanie jest częstokroć niebanalne. W tym rozdziale będziemy musieli nieźle kombinować!

Historycznie duży wpływ na rozwój kombinatoryki miały gry hazardowe i potrzeba szacowania prawdopodobieństwa wygranej, od czego zależała strategia gry. W 1654 r. Antoine Gombaud¹, zadziwiony pozorną sprzecznością w pewnej grze w kości, rozpoczął długotrwałą korespondencję z Pascalem², który następnie komunikował się z Fermatem³, co dało początki probabilistyki. Konkretnie problem Gombauda był następujący: w kasynie rzucamy 24 razy pod rząd dwiema

¹ Antoine Gombaud, kawaler de Méré (1607-1684), francuski pisarz.

² Blaise Pascal (1623-1662), francuski matematyk, fizyk i filozof, współtwórca rachunku prawdopodobieństwa, m.in. sformułował zasadę indukcji matematycznej.

³ Pierre de Fermat (1601–1665), francuski matematyk, przyczynił się do rozwoju geometrii analitycznej, teorii liczb i rachunku prawdopodobieństwa.

kościmi do gry. Jeśli choć raz wypadnie podwójna jedynka, wygrywamy. Czy mamy obstawiać? Odpowiedź na to pytanie jest prosta, jeśli przestudiujemy kolejne rozdziały (zob. zadanie 2.1).

Ponieważ podstawowe pojęcia kombinatoryczne oraz podstawy rachunku prawdopodobieństwa uczone są w szkole średniej, w tym wykładzie skoncentrujemy się na bardziej zaawansowanych aspektach tych tematów.

2.1 Permutacje

Z *permutacjami* studenci spotykają się namacalnie, stojąc w kolejce w dziekanacie po wpis warunkowy z matematyki dyskretniej! Mówiąc luźno, każda kolejka to permutacja stojących w niej ludzi. Podamy teraz formalną definicję.

Def. 2.1 (permutacja). *Permutacją n obiektów tworzących zbiór S nazywamy każdą funkcję ze zbioru n kolejnych liczb naturalnych, $\{1, 2, \dots, n\}$, na zbiór S .*

Innymi słowy, każdemu elementowi zbioru S nadajemy numer, czyli *ustawiamy obiekty w kolejności*. Tak właśnie najłatwiej zobrazować permutację. Jeszcze inaczej: tworzymy *ciąg n -wyrazowy* ze wszystkich elementów zbioru S .

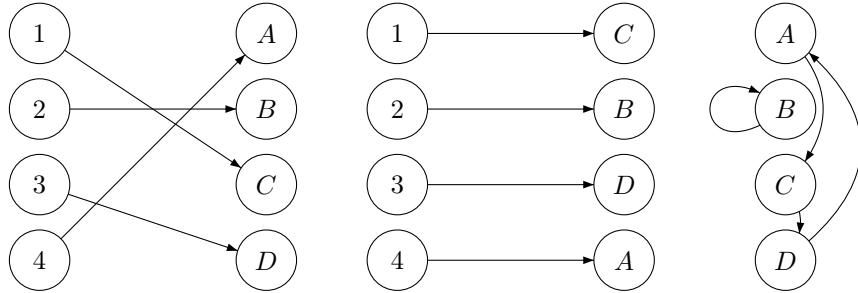
Przykład 2.1. *Rozważmy zbiór czteroelementowy $S = \{A, B, C, D\}$. Permutacją jest funkcja $f : \{1, 2, 3, 4\} \rightarrow S$ zadaną np. jako $f(1) = C$, $f(2) = B$, $f(3) = D$, $f(4) = A$. Wygodnie jest przyjąć konwencję, że wypisujemy elementy od lewej do prawej według kolejności rosnącego porządku liczb naturalnych, tak więc podaną właśnie przykładową permutację możemy zapisać po prostu jako $CBDA$, por. lewa i środkowa część rys. 2.1.*

Ponieważ zbiory o tej samej liczbie elementów n możemy ze sobą utożsamić (np. możemy się umówić, że jedynce odpowiada A , dwójce B itd.), w literaturze stosuje się jeszcze inną definicję permutacji:

Uwaga 2.1. *Permutacją n obiektów tworzących zbiór S nazywamy każde odwzorowanie zbioru S na siebie.*

Definicja ta zobrazowana jest na prawej części rys. 2.1, który należy odczytywać w następujący sposób: „naturalną”, czy wyjściową kolejnością jest $ABCD$. Idziemy za strzałkami na rysunku i w miejscu A wstawiamy C , w miejscu B – B , w miejscu C – D , wreszcie w miejscu D – A , co daje permutację $CBDA$.

Warto się raz zmóc z jawnym wypisaniem permutacji, co nawet dla małych wartości n jest czasochłonne. Wszystkie permutacje czteroelementowego zbioru



Rysunek 2.1: Graficzna ilustracja przykładowej permutacji zbioru czteroelementowego. Wszystkie trzy diagramy przedstawiają permutację CBDA. Lewy diagram odpowiada definicji (2.1), środkowy diagram przedstawia to samo z „rozsupłanymi” strzałkami, uwidoczniając kolejność CBDA, natomiast prawy diagram odpowiada równoważnej definicji (2.1)

$\{A, B, C, D\}$ to następujące 24 ciągi:

$$\begin{array}{cccccccc}
 \text{ABCD} & \text{ABDC} & \text{ACBD} & \text{ACDB} & \text{ADBC} & \text{ADCB} \\
 \text{BACD} & \text{BADC} & \text{BCAD} & \text{BCDA} & \text{BDAC} & \text{BDCA} \\
 \text{CABD} & \text{CADB} & \text{CBAD} & \text{CBDA} & \text{CDAB} & \text{CDBA} \\
 \text{DABC} & \text{DACB} & \text{DBAC} & \text{DBCA} & \text{DCAB} & \text{DCBA} .
 \end{array} \tag{2.1}$$

Podstawową funkcją całkowitoliczbową w kombinatoryce jest poznana już wcześniej *silnia*. Poznamy teraz więcej własności tej funkcji oraz jej związek z liczbą permutacji i innymi wielkościami kombinatorycznymi. Wprowadźmy najpierw symbol iloczynu:

Def. 2.2. $\Pi_{i=1}^n a_i = a_1 \cdot a_2 \cdot \dots \cdot a_n$.

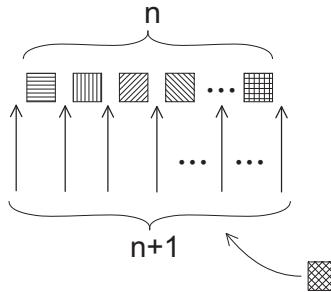
Def. 2.3. Silnią nazywamy funkcję określoną na zbiorze liczb naturalnych w następujący sposób:

$$n! = 1 \cdot 2 \cdot \dots \cdot n = \Pi_{i=1}^n i, \quad n \geq 1,$$

czyli jest to iloczyn n kolejnych liczb naturalnych.

Tw. 2.1. [liczba permutacji] Oznaczmy liczbę wszystkich różnych permutacji n -elementowego zbioru jako P_n . Zachodzi $P_n = n!$

Dowód: (indukcja matematyczna) Dla $n = 1$ mamy w oczywisty sposób $P_1 = 1$. Następnie założymy, że mamy n obiektów ustawionych w kolejności. Mogliśmy to, na mocy założenia indukcyjnego, zrobić na $P_n = n!$ sposobów. Teraz dokładamy obiekt $n + 1$, który możemy ustawić na początku, w $n - 1$ miejscach pomiędzy ustawionymi już n obiektami oraz na końcu, co łącznie daje $n + 1$ możliwości. Te $(n + 1)$ możliwości mamy dla każdego ułożenia n obiektów, a zatem otrzymujemy $P_{n+1} = (n + 1)P_n = (n + 1)!$ \square



Rysunek 2.2: Idea dowodu indukcyjnego, pokazującego, iż liczba permutacji zbioru n -elementowego wynosi $n!$. Poszczególne n elementów zbioru ustawiono w pewnej kolejności. Kolejny element można wstawić w miejsca wskazane strzałkami: na początku, na końcu oraz w $n - 1$ pozycji pomiędzy n elementami, co łącznie daje $n + 1$ możliwości

Idea dowodu przedstawiona jest na rys. 2.2. W szczególności $4! = 24$, czego możemy doliczyć się w tabeli (2.1) zawierającej wszystkie permutacje zbioru czteroelementowego. Alternatywny dowód Tw. 2.1 jest następujący:

Dowód: Ustawiając n elementów w kolejce, pierwszy możemy wybrać na n sposobów, drugi na $(n - 1)$ sposobów, trzeci na $(n - 2)$ sposoby, ..., n -ty na jeden sposób. Łącznie daje to $n!$ możliwości. \square

Zastanówmy się jeszcze, na ile sposobów możemy ustawić zero obiektów, czyli ile wynosi P_0 . Może się to wydać czysto akademickim gdybaniem i autorzy zazwyczaj rozprawiają się z tym problemem lakonicznym stwierdzeniem, że *przyjmujemy konwencję* $0! = 1$. Okaże się, że taka konwencja jest w istocie bardzo przydatna dla zwartej notacji w praktycznych obliczeniach. Spróbujmy jednak podać kilka argumentów. Najpierw zastosujmy rekurencję otrzymaną w powyższym dowodzie, $P_{n+1} = (n + 1)P_n$, do tyłu, czyli

$$P_n = \frac{P_{n+1}}{n+1}.$$

Wynika stąd natychmiast, że $P_0 = 1$. Jeślibyśmy jednak chcieli pociągnąć tę „rekurencję wspak” jeszcze dalej i otrzymały P_{-1} , to napotkamy na problem, bo $P_{-1} = P_0/0 = \infty$. Z pomocą w zrozumieniu, co prawda nie kombinatorycznym, przychodzi tutaj analiza matematyczna i funkcja Γ Eulera⁴, będąca uogólnieniem silni na zbiór liczb zespolonych i zdefiniowana jako

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt. \quad (2.2)$$

⁴ Leonhard Euler (1707-1783), szwajcarski matematyk i fizyk, jeden z największych matematycznych geniuszy wszech czasów, twórca ponad 900 dzieł matematycznych, w których bardzo istotnie przyczynił się do rozwoju wszystkich znanych w XVIII w. działów matematyki, jeden z ojców współczesnej matematyki.

Mamy przy tym (zob. ćw. 2.1)

$$n! = \Gamma(n+1). \quad (2.3)$$

Wyrażenie (2.2) ma sens poza $z = 0, -1, -2, \dots$ (dla wtajemniczonych: funkcja $\Gamma(z)$ ma w tych miejscach bieguny). Jednocześnie $0! = \Gamma(1) = \int_0^\infty e^{-t} dt = 1$, podobnie, jak uzyskaliśmy powyżej z rekurencji wspak.

Zgodnie z powyższymi argumentami przyjmujemy

Def. 2.4. $0! = 1$,

co kombinatorycznie oznacza, że zero obiektów można uporządkować na jeden sposób.

2.2 Silnia

Teraz zajmiemy się oszacowaniem silni dla dużych wartości n . W tym celu skorzystamy z prostych metod analizy matematycznej, zarazem ilustrując ich użyteczność w zagadnieniach kombinatorycznych. Wiemy już, że $n!$ rośnie bardzo szybko, oto 12 pierwszych wyrazów poczawszy od $0!$:

$$1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800, 479001600, \dots$$

Oszacowanie za pomocą prostej formuły, które za chwilę wyprowadzimy, ma istotne znaczenie praktyczne, podobne do rozwiązania rekurencji. Silnia stanowi bowiem rozwiązanie ciągu rekurencyjnego $P_{n+1} = (n+1)P_n$ i obliczenie n -tego wyrazu wymaga wykonania n mnożeń z coraz większymi liczbami, co jest niezwykle pracochłonne. Niestety, rekurencja ta nie należy do klas z rozdz. 1, które już umiemy rozwiązywać, zatem potrzebny jest tu inny sposób.

Pierwsza metoda oparta jest na całkowaniu metodą trapezów. Napiszmy najpierw, logarytmując,

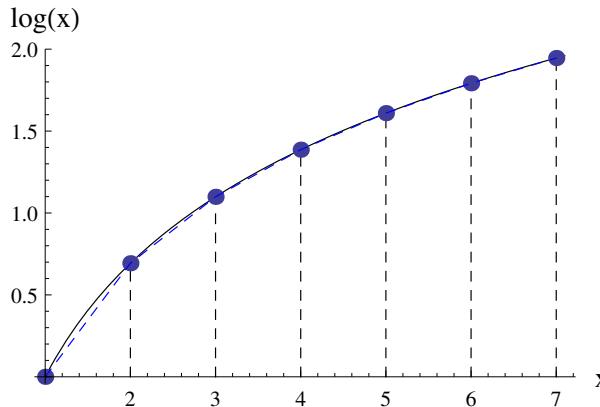
$$\log n! = \log(1 \cdot 2 \cdot 3 \dots n) = \log 1 + \log 2 + \log 3 + \dots + \log n.$$

Z drugiej strony, obliczmy całkę

$$I_n = \int_1^n \log x \, dx = (x \log x - x)|_1^n = n \log n - n + 1. \quad (2.4)$$

Całkę tę możemy następnie przybliżyć, stosując regułę trapezów, co przedstawia poglądowo rys. 2.3. Pole trapezu o numerze i , licząc od lewej strony wynosi, $t_i = \frac{1}{2}[\log i + \log(i+1)]$. A zatem przybliżenie dla całki (2.4) na przedziale $[1, n]$ jest następujące:

$$\begin{aligned} I_n &\simeq t_1 + t_2 + \dots + t_{n-1} = \\ &= \frac{1}{2}(\log 1 + \log 2) + \frac{1}{2}(\log 2 + \log 3) + \dots + \log(n-1)) + \frac{1}{2}(\log(n-1) + \log n) \\ &= \frac{1}{2} \log 1 + \log 2 + \log 3 + \dots + \log(n-2) + \log(n-1) + \frac{1}{2} \log n, \end{aligned}$$



Rysunek 2.3: Obliczanie całki (2.4) w sposób dokładny, jako pole figury pod linią ciągłą, oraz w sposób przybliżony, jako suma pól trapezów zaznaczonych liniami przerywanymi

(oczywiście, pisany dla symetrii $\log 1 = 0$). A zatem,

$$I_n = \log n! - \frac{1}{2} \log n.$$

Porównując to wyrażenie z wynikiem całki (2.4), dostajemy

$$\log n! - \frac{1}{2} \log n \simeq n \log n - n + 1,$$

W wyrażeniu tym mamy przybliżoną równość, albowiem dokładna wartość całki i jej przybliżenie metodą trapezów (zob. rys. 2.3) nie są sobie równe. Możemy oszacować różnicę na odcinku $[i, i+1]$, oznaczoną jako d_i :

$$d_i = \int_i^{i+1} \log x \, dx - t_i = \left(i + \frac{1}{2} \right) \log \frac{i+1}{i} - 1 = \mathcal{O}(1/i^2).$$

Ponieważ d_i maleje dla dużych i jak $1/i^2$, suma $\sum_{i=1}^n d_i$ jest zbieżna dla $n \rightarrow \infty$ do pewnej stałej c (z rys. 2.3 wynika, że $c > 0$). Tak więc⁵

$$\log n! \sim n \log n - n + 1 + \frac{1}{2} \log n - c.$$

Wprowadzając $C = e^{1-c}$ i „delegarytmując” powyższe wyrażenie, otrzymujemy oszacowanie

$$n! \sim C \sqrt{n} e^{-n} n^n. \quad (2.5)$$

Uzyskaliśmy więc, za pomocą bardzo prostej metody, asymptotyczną postać $n!$, z wyjątkiem dokładnego wyznaczenia stałej C .

Aby uzyskać wzór asymptotyczny dla $n!$ z wyznaczoną wartością stałej C , użyjemy funkcji Γ Eulera (2.2) i związku

$$n! = \int_0^\infty dt t^n e^{-t} = n^{n+1} \int_0^\infty dy y^n e^{-ny} = n^{n+1} e^{-n} \int_0^\infty dy e^{n(\log y - y + 1)}, \quad (2.6)$$

⁵ Używamy tu symbolu $a_n \sim b_n$ w znaczeniu $\lim_{n \rightarrow \infty} a_n/b_n = 1$.

gdzie dokonaliśmy podstawienia $t = ny$, a w ostatniej równości wyciągnęliśmy e^{-n} przed całkę. Funkcja podcałkowa $e^{n(\log y - y + 1)}$ ma pewne ważne cechy zilustrowane na rys. (2.4). Niezależnie od wartości n posiada maksimum w $y = 1$, które przy wzrastającym n staje się coraz węższe. Wykładnik $\log y - y$ ma bowiem maksimum w $y = 1$. W związku z tym, aby obliczyć (bardzo dokładnie) całkę (2.6) dla dużych n , potrzebujemy znać funkcję podcałkową jedynie w bezpośrednim sąsiedztwie $y = 1$. Wprowadźmy więc podstawienie $y = 1 + \xi$ i skorzystajmy z rozwinięcia Taylora wokół $\xi = 0$ dla logarytmu:

$$\log(1 + \xi) = \xi - \frac{1}{2}\xi^2 + \frac{1}{3}\xi^3 - \dots$$

Otrzymujemy wówczas z (2.6) wzór

$$n! = n^{n+1}e^{-n} \int_{-1}^{\infty} d\xi e^{n(\xi - \frac{1}{2}\xi^2 + \dots - 1 - \xi + 1)} = n^{n+1}e^{-n} \int_{-1}^{\infty} d\xi e^{-\frac{n}{2}\xi^2 + \dots},$$

gdzie kropki oznaczają człony rzędu ξ^3 i wyższe. Można pokazać, że człony te wnoszą poprawki rzędu względnego $1/n$ do wyniku, więc zostaną w naszej analizie pominięte. Ponadto, dolną granicę całkowania można rozszerzyć do $-\infty$, ponieważ (por. rys. 2.4) obszar $y < 0$ wnosi zaniedbywalny wkład do całki dla dużych wartości n . Mamy zatem

$$n! \sim n^{n+1}e^{-n} \int_{-\infty}^{\infty} d\xi e^{-\frac{n}{2}\xi^2}.$$

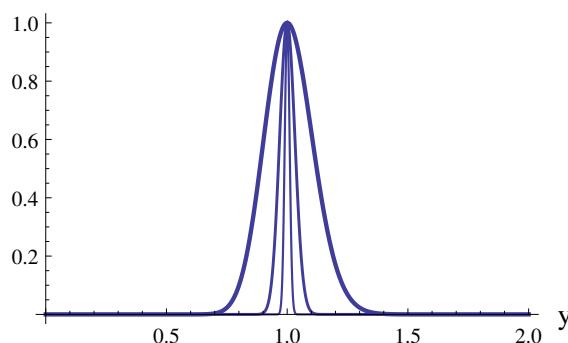
Korzystając teraz z wzoru na całkę gaussowską

$$\int_{-\infty}^{\infty} d\xi e^{-\frac{\xi^2}{a}} = \sqrt{2\pi a},$$

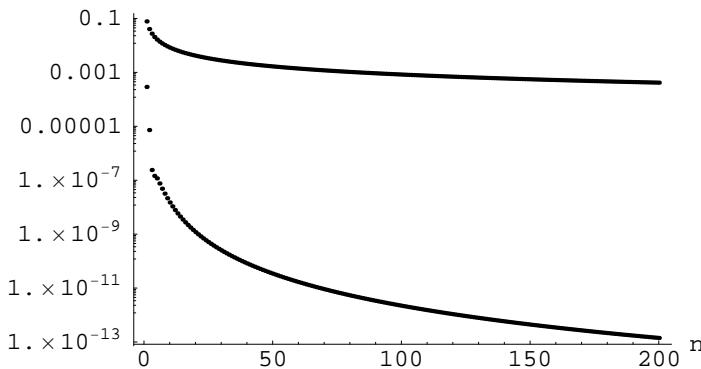
uzyskujemy ostateczny wynik:

$$n! \sim \sqrt{2\pi n} e^{-n} n^n. \quad (2.7)$$

Exp[n(log(y)-y+1)]



Rysunek 2.4: Funkcja podcałkowa z wzoru (2.6) dla $n = 100, 1000$ i 10000 . Kolejnymi wartościami odpowiadają coraz węższe krzywe



Rysunek 2.5: Błąd względny przybliżenia $n!$ ze wzoru Stirlinga narysowany jako funkcja n . Wykres górny – zwykły wzór Stirlinga (2.7), wykres dolny – wzór Stirlinga z poprawkami (2.9)

Jest to tzw. *wzór Stirlinga*⁶. Formułę tę trzeba zapamiętać, bo będzie wielokrotnie przydatna! Porównując wzór Stirlinga z wyrażeniem (2.5) widzimy, że $C = \sqrt{2\pi}$.

Z pomocą bardziej zaawansowanej analizy można wyprowadzić wzór

$$\sqrt{2\pi}ne^{-n}n^n e^{\frac{1}{12n+1}} \leq n! \leq \sqrt{2\pi}ne^{-n}n^n e^{\frac{1}{12n}}. \quad (2.8)$$

Jeszcze dokładniejsza postać wzoru Stirlinga, otrzymana za pomocą tzw. rozwińcia asymptotycznego, to *wzór Stirlinga z poprawkami*:

$$n! \sim \sqrt{2\pi}ne^{-n}n^n \left(1 + \frac{1}{12n} + \frac{1}{288n^2} - \frac{139}{51840n^3} + \dots \right). \quad (2.9)$$

Chociaż formalnie wzór Stirlinga stosuje się dla dużych n , to jest on niezwykle dokładny nawet dla małych n (z wyjątkiem $n = 0$). W szczególności przybliżenie (2.9) daje 0.9997 dla $n = 1$, 1.99999 dla $n = 2$, a im dalej tym jeszcze lepiej. Dla $n = 20$ dokładność względna jest na poziomie 10^{-9} . Do szybkich oszacowań wystarcza z powodzeniem zwykły wzór Stirlinga (2.7), którego dokładność dla $n = 20$ jest na poziomie promili. Rysunek 2.5 przedstawia błąd względny obu przybliżeń, tj. wielkość $|wynik dokładny - przybliżenie|/(wynik dokładny)$ w funkcji n .

Teraz wprowadzimy symbol tzw. podwójnej silni. Wbrew pozorom nie jest to silnia silni, co oznaczylibyśmy jako $(n!)!$, ale znacznie mniej!

Def. 2.5.

$$(2k-1)!! = 1 \cdot 3 \cdot 5 \cdots (2k-1),$$

$$(2k)!! = 2 \cdot 4 \cdot 6 \cdots 2k, \quad k = 1, 2, 3 \dots$$

Dla n nieparzystego, $n = 2k-1$, jest to zatem iloczyn kolejnych liczb nieparzystych aż do n , a dla n parzystego, $n = 2k$, iloczyn kolejnych liczb parzystych.

⁶ James Stirling (1692-1770), szkocki matematyk.

Jawne rozpisanie daje natychmiast $n! = n!!(n - 1)!!$, ponadto dla parzystego n mamy związek $n!! = 2^{n/2}(n/2)!!$. Na przykład $8! = 2 \cdot 4 \cdot 6 \cdot 8 = 2^4 4!$.

Typowym przykładem zastosowania permutacji są zliczania liczby ustawień obiektów.

Przykład 2.2. *Policzmy, na ile różnych sposobów możemy posadzić n gości przy okrągłym stole, przy czym istotne jest tylko to, kto z kim sąsiaduje. Inaczej mówiąc, traktujemy jako równoważne ustawienia uzyskane poprzez obroty, gdzie każdy przesuwa się o tyle samo miejsc w lewo bądź w prawo, oraz ustawienia uzyskane poprzez zamianę orientacji „zgodnie z ruchem wskazówek zegara” na kolejność „odwrotnie do ruchu wskazówek zegara” – ta zmiana powoduje, że sąsiad z lewej strony staje się sąsiadem po prawej. Wszystkich ustawień, bez podanych równoważności, byłoby $n!$. Dzielimy przez liczbę obrotów n , następnie przez liczbę orientacji równą 2. Tak więc szukana liczba ustawień gości to $(n-1)!/2$. Widzimy teraz, że dla $n = 1$ i $n = 2$ coś tu nie pasuje, bo dostajemy bezsensowny wynik $1/2$. Dla tych wartości nie mamy bowiem dwóch różnych orientacji, więc w tym przypadku nie dzielimy przez 2. Ten przykład pokazuje ku przestrodze, że nie możemy w tego typu zadaniach „bezkrytycznie” stosować uzyskanych wzorów dla każdego n . Początkowe wyrazy uzyskanego ciągu to $1, 1, 1, 3, 12, 60, 360, \dots$*

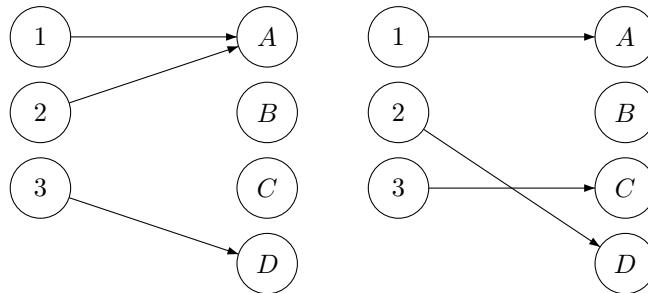
2.3 Wariacje

W poprzednim podrozdziale zdefiniowaliśmy formalnie permutację n obiektów tworzących zbiór S jako funkcję ze zbioru $\{1, 2, \dots, n\}$ na zbiór S . Specyfikacja na zbiór była istotna. Mieliśmy więc suriekcję, a ze względu na równoliczność zbiorów także iniekcję, co łącznie dawało bijekcję, patrz rys. 2.1. Zrezygnujemy teraz z tych ograniczeń i rozważymy odwzorowanie ze zbioru $\{1, 2, \dots, n\}$ w m -elementowy zbiór S .

Def. 2.6 (wariacja z powtórzeniami). Funkcję ze zbioru $\{1, 2, \dots, n\}$ w m -elementowy zbiór S nazywamy n -elementową wariacją z powtórzeniami m -elementowego zbioru S .

Przykład 2.3. Rozważmy ponownie zbiór czteroelementowy $S = \{A, B, C, D\}$. Przykładem wariacji trójelementowej z powtórzeniami tego zbioru jest funkcja $f : \{1, 2, 3\} \rightarrow S$ zadana jako $f(1) = A, f(2) = A, f(3) = D$. Ponownie przyjmujemy konwencję, że wypisujemy elementy od lewej do prawej według kolejności rosnącego porządku liczb naturalnych, zatem nasza wariacja z powtórzeniami to AAD , por. rys. 2.6.

Wszystkie trójelementowe wariacje z powtórzeniami czteroelementowego zbioru $\{A, B, C, D\}$ są następujące:



Rysunek 2.6: Graficzna ilustracja przykładowej trójelementowej wariacji z powtórzeniami, AAD (lewy diagram) i bez powtórzeń, ADC (prawy diagram) zbioru czteroelementowego $\{A, B, C, D\}$

AAA	AAB	AAC	AAD	ABA	ABB	ABC	ABD
ACA	ACB	ACC	ACD	ADA	ADB	ADC	ADD
BAA	BAB	BAC	BAD	BBA	BBB	BBC	BBD
BCA	BCB	BCC	BCD	BDA	BDB	BDC	BDD
CAA	CAB	CAC	CAD	CBA	CBB	CBC	CBD
CCA	CCB	CCC	CCD	CDA	CDB	CDC	CDD
DAA	DAB	DAC	DAD	DBA	DBB	DBC	DBD
DCA	DCB	DCC	DCD	DDA	DDD	DDC	DDD

Uff ...

Tw. 2.2. *Liczba wszystkich wariacji z powtórzeniami (2.6) wynosi m^n .*

Dowód: (indukcja matematyczna) Dla $n = 1$ mamy w oczywisty sposób m możliwości, bo wybieramy jeden z m elementów zbioru S . Założenie indukcyjne mówi, że dla n mamy m^n możliwości. Kiedy zwiększymy n o jeden, dla każdej z tych istniejących możliwości możemy dobrać nowy element na m sposobów, a więc mamy teraz m^{n+1} , co kończy dowód. \square

Przykład 2.4. *Zapis co najwyżej k -cyfrowej liczby w systemie dziesiętnym jest przykładem k -elementowej wariacji z powtórzeniami dziesięcioelementowego zbioru cyfr $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Zgodnie z Tw. 2.2 mamy tu łącznie 10^k możliwości. Istotnie, co najwyżej 3-cyfrowych liczb jest, łącznie z zerem, $10^3 = 1000$.*

Zauważmy, że n -elementowych wariacji z powtórzeniami n -elementowego zbioru jest n^n , czyli więcej niż liczba permutacji $n!$. Jest to oczywiste, bo permutacje podlegały ograniczeniu – nie mogliśmy powtarzać elementów. Ten fakt jest też widoczny we wzorze Stirlinga (2.7). Pierwszych kilka elementów ciągu n^n to

Def. 2.7 (wariacja bez powtórzeń). Różnowartościową funkcję ze zbioru $\{1, 2, \dots, k\}$ w n -elementowy zbiór S nazywamy k -elementową wariacją bez powtórzeń n -elementowego zbioru S , przy czym $n \geq k$.

Oczywiście n musi być większe lub równe k , aby funkcja mogła być różnowartościowa.

Przykład 2.5. Weźmy jeszcze raz nasz zbiór $S = \{A, B, C, D\}$. Przykładem wariacji trójelementowej bez powtórzeń tego zbioru jest funkcja $f : \{1, 2, 3\} \rightarrow S$ zadana jako $f(1) = A, f(2) = D, f(3) = C$, co zapisujemy ADC , por. rys. 2.6.

Tw. 2.3. Liczba wszystkich wariacji bez powtórzeń (2.7) wynosi

$$V_k^n = n(n-1)(n-2) \cdots (n-k+1) = \frac{n!}{(n-k)!}.$$

Dowód: Pierwszy element możemy wybrać na n sposobów, drugi już tylko na $(n-1)$, bo nie możemy powtórzyć pierwszego wybranego elementu, trzeci na $n-2$ sposobów itd., k -ty na $n-k+1$ sposobów, co łącznie daje powyższy wzór na V_k^n . \square

Często stosuje się oznaczenie

Def. 2.8 (symbol Pochhammerra⁷).

$$(x)_k = x(x+1)(x+2) \cdots (x+k-1),$$

będący iloczynem k członów, następny większy o 1 od poprzedniego. Proste porównanie daje $V_k^n = (n-k+1)_k$. Zauważmy też, że dla $n = m$ wariacje bez powtórzeń są tym samym, co permutacje. Wówczas $V_n^n = n!/0! = n!$, tak, jak powinno być. Przydał się nam teraz fakt, że $0! = 1$.

2.4 Permutacje z powtórzeniami

Teraz kolejna definicja do kompletu:

Def. 2.9. n -elementową permutację z powtórzeniami o krotnościach n_1, n_2, \dots, n_m m -elementowego zbioru S nazywamy n -elementową wariację z powtórzeniami zbioru S , w której element j jest przyporządkowany n_j razy.

Przykład 2.6. Niech $S = \{A, B, C, D\}$. Trójelementowe permutacje z powtórzeniami o krotnościach 1, 2, 0, 0 są następujące: ABB, BAB, BBA .

⁷ Leo August Pochhammer (1841-1920), pruski matematyk.

Tw. 2.4. *Liczba permutacji z powtórzeniami o krotnościach n_1, \dots, n_m wynosi*

$$P_{n_1 n_2 \dots n_m}^n = \frac{n!}{n_1! n_2! \dots n_m!},$$

gdzie $n = n_1 + n_2 + \dots + n_m$.

Dowód: Gdybyśmy rozróżniali wszystkie powtarzane elementy, to liczba wszystkich możliwości byłaby równa liczbie permutacji n -elementowego zbioru, tj. $n!$. Elementy j zbioru S przyjmowałyby wtedy $n_j!$ możliwych ustawić. Ponieważ jednak powtarzanych elementów nie rozróżniamy, utożsamiamy te ustawienia, a więc musimy podzielić $n!$ przez $n_j!$ dla każdego $j = 1, \dots, m$. \square

Przykład 2.7. Wracając do poprzedniego przykładu, gdybyśmy rozróżniali element B jako B_1 i B_2 , to mielibyśmy $3! = 6$ możliwości AB_1B_2 , AB_2B_1 , B_1AB_2 , B_2AB_1 , B_1B_2A , B_2B_1A . Ponieważ utożsamiamy B_1 i B_2 , musimy wydzielić przez możliwe $2!$ ustawień B_1 i B_2 , zatem w wyniku mamy $3!/2! = 3$ możliwości z przykładu 2.6.

Dodajmy jeszcze, że krotności niewystępujących elementów wynoszą 0, ale ponieważ $0! = 1$, możemy sobie do woli dzielić przez $0!$.

Z definicji 2.9 wynika, że suma permutacji z powtórzeniami po wszystkich możliwych krotnościach n_i z warunkiem $n_1 + \dots + n_m = n$, daje wszystkie n -elementowe wariacje z powtórzeniami m -elementowego zbioru. W związku z tym otrzymujemy nietrywialnie wyglądającą tożsamość

$$\sum_{\substack{n_1, \dots, n_m=0 \\ n_1+\dots+n_m=n}}^n \frac{n!}{n_1! \dots n_m!} = m^n. \quad (2.10)$$

Druga linijka pod znakiem sumy specyfikuje warunek nałożony na wskaźniki sumowania.

2.5 Kombinacje

W wariacjach bez powtórzeń w oczywisty sposób istotna była kolejność, np. wariacje ACD , DAC , CDA , CAD , DCA , ADC są różne. Jeśli kolejność występowania elementów nie jest istotna, to mówimy o kombinacjach.

Def. 2.10 (kombinacja). *k -elementową kombinacją n -elementowego zbioru S nazywamy k -elementowy podzbiór zbioru S .*

W odróżnieniu od poprzednio zdefiniowanych wielkości kombinatorycznych, które były funkcjami, kombinacja nie jest funkcją, tylko podzbiorem.

Przykład 2.8. Dla zbioru $S = \{A, B, C, D\}$ trójelementowymi kombinacjami są podzbiory $\{A, B, C\}$, $\{A, B, D\}$, $\{A, C, D\}$ i $\{B, C, D\}$. Inspekcja pokazuje, że są to wszystkie możliwości.

Def. 2.11 (symbol Newtona). Dla całkowitych $n \geq 0$ i k

$$\binom{n}{k} = \begin{cases} \frac{n!}{k!(n-k)!} & \text{dla } 0 \leq k \leq n \\ 0 & \text{dla } k < 0 \text{ lub } k > n \end{cases}$$

(czytamy „ n po k ”).

Tw. 2.5. Liczba kombinacji (2.10) wynosi

$$C_k^n = \binom{n}{k}.$$

Dowód: Zaczniemy od utworzenia k -elementowych wariacji bez powtórzeń n -elementowego zbioru, których jest V_k^n . Nie zwracanie uwagi na kolejność oznacza, że utożsamiamy każdą wariację zawierającą te same elementy. Wariacji bez powtórzeń zawierających te same elementy w różnej kolejności jest $k!$, dzielimy więc V_k^n przez $k!$ i dostajemy

$$C_k^n = \frac{V_k^n}{k!} = \frac{n!}{(n-k)!k!} = \binom{n}{k}.$$

□

W szczególności trójelementowych kombinacji czteroelementowego zbioru jest 4, tyle, ile w przykładzie 2.8. Zauważmy też, że zgodnie z naszą konwencją jest jedna zeroelementowa kombinacja zeroelementowego zbioru, bo $\binom{0}{0} = 1$. Czyli zbiór pusty ma jeden podzbiór, mianowicie zbiór pusty! Wszystko się zgadza.

Inny sposób spojrzenia na liczbę kombinacji jest następujący: odwzorujmy n -elementowy zbiór S w dwuelementowy zbiór $\{0, 1\}$, w ten sposób znakując elementy, które należą do danego podzbioru przez 1, a te, które doń nie należą, przez 0. Widać wówczas, że liczba k -elementowych kombinacji n -elementowego zbioru jest równa liczbie n -elementowych permutacji z powtórzeniami o krotnościach k (elementy należące do podzbioru) oraz $n - k$ (elementy nienależące do podzbioru). Wobec tego

$$C_k^n = P_{k,n-k}^n,$$

co oczywiście jest prawdą. Z kolei wracając do permutacji z powtórzeniami, zauważamy, że P_{n_1, \dots, n_m}^n można interpretować jako liczbę wszystkich podziałów

n -elementowego zbioru na m rozłącznych podzbiorów o krotnościach n_1, \dots, n_m . Tę własność warto zapamiętać!

Podamy teraz kilka najprostszych własności symbolu Newtona. Wprost z definicji wynika, że

Tw. 2.6. *Dla całkowitych $n \geq 0$ i k*

$$\binom{n}{n} = \binom{n}{0} = 1,$$

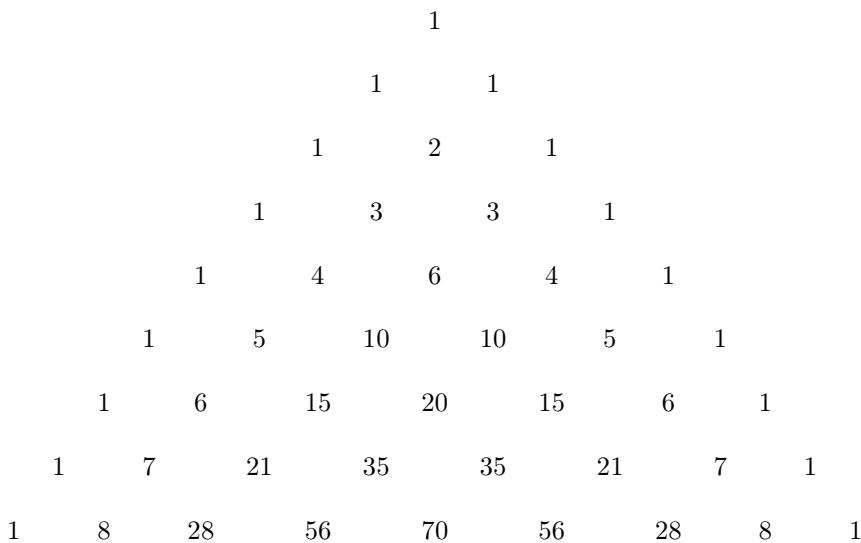
$$\binom{n}{k} = \binom{n}{n-k}, \quad (2.11)$$

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}. \quad (2.12)$$

Dowód: Dwie pierwsze równości są trywialne. W (2.12) sprowadzamy lewą stronę do wspólnego mianownika, co daje

$$\begin{aligned} \frac{n!}{k!(n-k)!} + \frac{n!}{(k+1)!(n-k-1)!} &= \frac{n!(k+1+n-k)}{(k+1)!(n-k)!} \\ &= \frac{(n+1)!}{(k+1)!((n+1)-(k+1))!}. \end{aligned}$$

□



Rysunek 2.7: Trójkąt Pascala

Rekurencja (2.12) prowadzi natychmiast do słynnego *trójkąta Pascala* [6] (znanego już wieki wcześniej Hindusom, Persom i Chińczykom!), gdzie n numeruje wiersze, a k kolumny (ukośnie, począwszy od 0). Każda liczba w tym trójkącie jest sumą dwóch liczb znajdujących się bezpośrednio nad nią. Efekt przedstawiony jest w całej okazałości na rys. 2.7. Puste miejsca na lewo i prawo od jedynek wypełnione są, zgodnie z definicją (2.11), zerami, których się nie wypisuje.

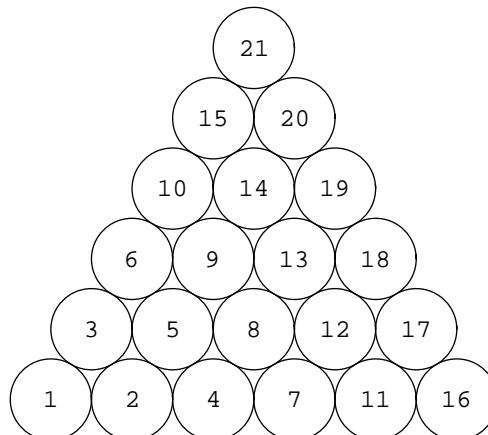
Trójkąt Pascala ma wiele magicznych własności. Na początku kolejnych *ukośnych* wierszy mamy jedynki, na pozycji drugiej są kolejne liczby naturalne, co nie jest jeszcze szczególnie fascynujące. Na trzecim miejscu są tzw. *liczby trójkątne*, czyli liczby jednogroszówk, które można ułożyć w ściśle upakowany trójkąt równoboczny jak na rys. 2.8. W następnym ukośnym wierszu mamy *liczby czworościenne* [6], tzn. liczby paciorek, z których można ułożyć ściśle upakowany czworościan foremny. Kolejne rzędy są uogólnieniami tej konstrukcji na większą liczbę wymiarów przestrzennych.

Zauważmy, że rekurencja (2.12) ma inny charakter niż rekurencje opisane w rozdz. 1. Rekurencje Hanoi czy Fibonacciego przebiegały w jednym wymiarze, tzn. ciągi miały jeden indeks n i rekurencja „poruszała” się wzduż tego indeksu. Natomiast rekurencja (2.12) przebiega w dwóch niezależnych wskaźnikach, n i k , jest więc dwuwymiarowa.

Zajmiemy się teraz bardzo użytecznymi własnościami symbolu Newtona.

Tw. 2.7. (*dwumian Newtona*)

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k}. \quad (2.13)$$



Rysunek 2.8: Liczby trójkątne: 1, 3, 6, 10, 15, 21, ..., widoczne w kolejnych wierszach w kółkach najbardziej na lewo. Liczby te tworzą trzeci rząd trójkąta Pascala, por. rys. 2.7

Dowód: (indukcja matematyczna) Dla $n = 0$ mamy $(a + b)^0 = 1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, natomiast

$$\begin{aligned} (a + b)^{n+1} &= (a + b)(a + b)^n = (a + b) \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} \\ &= \sum_{k=0}^n \binom{n}{k} a^{k+1} b^{n-k} + \sum_{k=0}^n \binom{n}{k} a^k b^{n-k+1} \\ &= \sum_{k=1}^{n+1} \binom{n}{k-1} a^k b^{n-k-1} + \sum_{k=0}^n \binom{n}{k} a^k b^{n-k+1} \\ &= \sum_{k=0}^{n+1} \left[\binom{n}{k-1} + \binom{n}{k} \right] a^k b^{n-k+1} = \sum_{k=0}^{n+1} \binom{n+1}{k} a^k b^{n-k+1}. \end{aligned}$$

W pierwszej sumie w trzeciej linijce przesunęliśmy wskaźnik sumowania, $k \rightarrow k - 1$, następnie skorzystaliśmy z definicji (2.11), by poszerzyć zakres sumowania, wreszcie użyliśmy rekurencji (2.12), co kończy dowód. \square

Wniosek 2.1. Podstawiając w Tw. 2.7 $a = b = 1$ oraz $a = 1, b = -1$, otrzymujemy

$$\sum_{k=0}^n \binom{n}{k} = 2^n, \quad (2.14)$$

$$\sum_{k=0}^n (-1)^k \binom{n}{k} = 0, \quad n > 0. \quad (2.15)$$

Istotnie, sumując liczby w kolejnych wierszach trójkąta Pascala, otrzymujemy kolejne potęgi dwójki, a sumując i jednocześnie zmieniając naprzemiennie znak, otrzymujemy zero. Inna użyteczna tożsamość wynika wprost z definicji:

$$\binom{n}{k} \binom{k}{j} = \binom{n}{j} \binom{n-j}{k-j}. \quad (2.16)$$

W zadaniu 2.8 zdefiniowany jest jeszcze jeden obiekt kombinatoryczny, tzw. *kombinacje z powtórzeniami*.

2.6 Współczynniki dwumianowe

Podamy teraz formalne uogólnienie symbolu Newtona, w którym górną liczbą jest dowolną liczbą rzeczywistą:

Def. 2.12 (współczynnik dwumianowy). Dla rzeczywistego x i całkowitego k

$$\binom{x}{k} = \begin{cases} \frac{x(x-1)\dots(x-k+1)}{k!} = \frac{(x-k+1)_k}{k!} & \text{dla } k \geq 0 \\ 0 & \text{dla } k < 0 \end{cases}.$$

Definicja daje więc dla $k \geq 0$ wielomian stopnia k w zmiennej rzeczywistej x . Wyrażenie to nie ma interpretacji kombinatorycznej, jednak jest ważne ze względu na liczne zastosowania.

Tw. 2.8. Dla dowolnego rzeczywistego p zachodzi

$$(a+b)^p = \sum_{k=0}^{\infty} \binom{p}{k} a^k b^{p-k}. \quad (2.17)$$

Dowód: Nie możemy tu skorzystać z własności kombinatorycznych, ale za pomocą przychodzi analiza matematyczna i szereg Taylora:

$$f(z) = f(0) + \frac{f'(0)}{1!}z + \frac{f''(0)}{2!}z^2 + \dots = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}z^k. \quad (2.18)$$

Oznaczmy $z = \frac{a}{b}$ i podzielmy obie strony wzoru (2.17) przez b^p , co daje

$$(1+z)^p = \sum_{k=0}^{\infty} \binom{p}{k} z^k. \quad (2.19)$$

Kolejne pochodne tego wyrażenia wynoszą

$$((1+z)^p)^{(k)} = p(p-1)\dots(p-k+1)(1+z)^{p-k},$$

zatem $((1+z)^p)^{(k)}|_{z=0} = p(p-1)\dots(p-k+1) = (p-k+1)_k$, skąd poprzez wzór Taylora wynika (2.19). \square

Zauważmy, że jeśli p jest nieujemną liczbą całkowitą, to suma po k urywa się na $k = p$. W przeciwnym przypadku musimy formalnie zsumować nieskończonie wiele członów. Wyrażenie (2.19) ma sens dla liczby zespolonej z , dla której $|z| < 1$. Istotnie, na mocy Tw. Cauchy'ego-Hadamarda promień zbieżności szeregu potęgowego (2.19) wynosi

$$\lim_{k \rightarrow \infty} \left(\frac{k!}{(p-k+1)_k} \right)^{1/k} = 1.$$

Wniosek 2.2. Ze wzoru dwumianowego (2.19) wynika natychmiast, że funkcja tworząca (1.9) dla współczynników dwumianowych ma postać $(1+z)^p$.

Na koniec tego podrozdziału podajemy jeszcze kilka użytecznych tożsamości dla współczynników dwumianowych, prawdziwych dla dowolnych rzeczywistych p, q oraz całkowitych j, k, m :

$$\binom{p}{k} = \binom{p-1}{k} + \binom{p-1}{k-1}, \quad (2.20)$$

$$\binom{p}{k} = \frac{p}{k} \binom{p-1}{k-1}, \quad (2.21)$$

$$\binom{p}{k} = (-1)^k \binom{k-p-1}{k}, \quad (2.22)$$

$$\binom{p}{j} \binom{j}{k} = \binom{p}{k} \binom{p-k}{j-k}, \quad (2.23)$$

$$\sum_{k=0}^j \binom{k}{m} = \binom{j+1}{m+1}, \quad (2.24)$$

$$\sum_{k=0}^j \binom{p+k}{k} = \binom{p+j+1}{j}, \quad (2.25)$$

$$\sum_{k=0}^j \binom{p}{k} \binom{q}{j-k} = \binom{p+q}{j}. \quad (2.26)$$

Ostatni wzór (2.26) nosi nazwę tożsamości Cauchy'ego. Elegancki dowód można podać opierając się na technice funkcji tworzących:

Dowód: Z (2.19) mamy

$$(1+z)^p = \sum_{l=0}^{\infty} \binom{p}{l} z^l, \quad (1+z)^q = \sum_{l=0}^{\infty} \binom{q}{l} z^l.$$

Mnożąc stronami i stosując wzór na iloczyn Cauchy'ego szeregów, otrzymujemy

$$(1+z)^{p+q} = \sum_{n=0}^{\infty} \sum_{k=0}^n \binom{p}{k} \binom{q}{n-k} z^n.$$

Natomiast z (2.19) otrzymujemy bezpośrednio rozwinięcie

$$(1+z)^{p+q} = \sum_{n=0}^{\infty} \binom{p+q}{n} z^n.$$

Porównanie współczynników przy potęgach z^n w dwóch powyższych wzorach kończy dowód. \square

2.7 Współczynniki wielomianowe

Powróćmy teraz do wzoru na liczbę permutacji z powtórzeniami:

$$P_{k_1, k_2, \dots, k_m}^n = \frac{n!}{k_1! k_2! \dots k_m!}, \quad n = k_1 + k_2 + \dots + k_m. \quad (2.27)$$

Oczywiście dla $n = 2$ powyższe wyrażenie redukuje się do symbolu Newtona (2.11). Uogólnieniem wzoru dwumianowego Newtona jest następujący pozyteczny wzór:

Tw. 2.9.

$$(z_1 + z_2 + \dots + z_m)^n = \sum_{k_1, \dots, k_m=0}^n P_{k_1, k_2, \dots, k_m}^n z_1^{k_1} \dots z_m^{k_m}, \quad n = k_1 + k_2 + \dots + k_m. \quad (2.28)$$

Dowód: Dowód przebiega przez indukcję względem m . Dla $m = 2$ wzór zachodzi na mocy Tw. (2.7). Założymy, że teza zachodzi dla m i rozważmy lewą stronę tezy dla $m + 1$:

$$\begin{aligned} L &= (z_1 + z_2 + \dots + z_m + z_{m+1})^n = [(z_1 + z_2 + \dots + z_m) + z_{m+1}]^n \\ &= \sum_{k=0}^n \binom{n}{k} (z_1 + z_2 + \dots + z_m)^k z_{m+1}^{n-k}, \end{aligned}$$

gdzie pogrupowaliśmy liczby z_i i skorzystaliśmy ze wzoru Newtona (2.7). Teraz wykorzystujemy założenie indukcyjne dla $(z_1 + z_2 + \dots + z_m)^k$, co daje

$$L = \sum_{k=0}^n \frac{n!}{k!(n-k)!} \sum_{k_1, \dots, k_m=0}^n \frac{k!}{k_1! k_2! \dots k_m! k_{m+1}!} z_1^{k_1} \dots z_m^{k_m} z_{m+1}^{n-k}.$$

Następnie upraszczamy $k!$ oraz zmieniamy indeks sumowania k na $k_{m+1} = n - k$,

$$L = \sum_{k_1, \dots, k_m, k_{m+1}=0}^n \frac{n!}{k_1! k_2! \dots k_m! k_{m+1}!} z_1^{k_1} \dots z_m^{k_m} z_{m+1}^{k_{m+1}},$$

co jest pożądaną prawą stroną tezy indukcyjnej. \square

2.8 Zasada szufladkowa

Podamy teraz bardzo użyteczne, choć na pierwszy rzut oka wyglądające nieco banalnie twierdzenie.

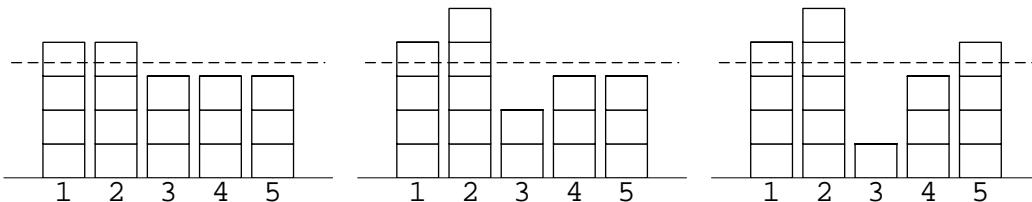
Tw. 2.10 (zasada szufladkowa Dirichleta). *Jeśli do k pudełek włożymy w dowolny sposób $n \geq 1$ przedmiotów, to istnieje pudełko, w którym jest co najmniej $\lceil n/k \rceil$, a także pudełko, w którym jest co najwyżej $\lfloor n/k \rfloor$ przedmiotów.*

Na przykład dla $n = 17$ i $k = 5$ mamy dla każdego podziału pudełko, w którym jest $\lceil 17/5 \rceil = 4$ lub więcej przedmiotów oraz pudełko, w którym jest $\lfloor 17/5 \rfloor = 3$ lub mniej przedmiotów.

Dowód: Wyobraźmy sobie najpierw, że rozdajemy n kart do gry pomiędzy k graczy. Możliwe są dwie sytuacje: jeśli n jest podzielne przez k , to rozdzielimy po równo i każdy gracz ma n/k kart, ponadto dla liczby całkowitej zachodzi $n/k = \lfloor n/k \rfloor = \lceil n/k \rceil$. Jeśli natomiast n nie jest podzielne przez k , to w którymś momencie braknie nam kart, żeby rozdać po równo i mamy związek $n = km + r$, gdzie m jest liczbą kart, którą otrzymali ci gracze, dla których zabrakło ostatniej karty, a reszta r jest liczbą graczy, którzy otrzymali $m + 1$ kart. Mamy zatem $m = n/k - r/k$. Ponieważ $0 < r < k$, zachodzą nierówności $n/k - 1 < m < n/k$, czyli $m = \lfloor n/k \rfloor$. Tak więc w wyniku rozdania $k - r$ graczy posiada $m = \lfloor n/k \rfloor$, a r graczy $m = \lfloor n/k \rfloor + 1 = \lceil n/k \rceil$ kart. Zatem w obu możliwych sytuacjach, całkowitego i niecałkowitego n/k , istnieją gracze, którzy mają $\lfloor n/k \rfloor$ kart, oraz gracze posiadający $\lceil n/k \rceil$ kart. Następny krok dowodu to wzajemne przekazywanie kart między uczestnikami. W wyniku takiej procedury gracz o najmniejszej liczbie kart może zmniejszyć swoje posiadanie, a gracz o największej liczbie kart zwiększyć je. Fakt, że w wyniku przekazywania można uzyskać dowolny rozkład, kończy dowód. \square

Idea zasady szufladkowej oraz jej dowodu przedstawiona jest na rys. 2.9.

Szczególny przypadek zasady szufladkowej mówi, że jeśli rozmieszczać $n > k$ obiektów w k szufladach, to w przynajmniej jednej szufladzie znajdują się co najmniej dwa obiekty. Angielska nazwa tej wersji to *pigeon-hole principle* – zasada gołębnika. Jeśli gołębnik ma k przegródek i mamy w nim $n > k$ gołębi, to musi istnieć przegródka, w której są co najmniej dwa gołębie. Stwierdzenie to brzmi nieco mniej banalnie, jeśli powiemy, że w Kielcach są przynajmniej dwie osoby o tej samej liczbie włosów (łysych pomijamy!) [40]. Stwierdzenie to



Rysunek 2.9: Zasada szufladkowa Dirichleta dla $n = 17$ obiektów rozmieszczanych w $k = 5$ pudełkach. W każdym przypadku istnieje pudełko zawierające $\lceil 17/5 \rceil = 4$ lub więcej przedmiotów oraz pudełko, w którym jest $\lfloor 17/5 \rfloor = 3$ lub mniej przedmiotów. Wynik „rozdawania” opisanego w dowodzie Tw. 2.10 zobrazowany jest na rysunku po lewej stronie. Linia przerywana oznacza iloraz $17/5 = 3.4$.

brzmi na pozór nieprawdopodobnie, ponieważ włosów na głowie jest bardzo dużo i „szansa” znalezienia dwóch osób o tej samej ich liczbie wydaje się nikła. Jednakże włosów na głowie mamy nie więcej niż 150000, a populacja Kielc to ponad 200000. Robimy szufladki o numerach 1–150000 i umieszczaemy w nich mieszkańców zgodnie z liczbą włosów na głowie. Co najmniej jedna szufladka będzie mieścić więcej niż jednego kielczanina! Dużo mniej oczywisty problem przedstawiony jest w ćw. 2.12.

2.9 Układanie domina

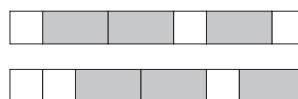
Czas na bardziej zaawansowany przykład zliczania kombinatorycznego. Ogólna grupa problemów ukazanych w tym podrozdziale nosi nazwę *układania domina* lub ogólniej *układania parkietu*: mamy pewien obszar, który musimy pokryć prostokątami, prostokątami oraz kwadratami itp. i pytamy o liczbę różnych pokryć. Jeden z najprostszych problemów tej klasy jest następujący:

Problem 2.1 (układanie domina i kwadratów w rzędzie). *Na ile sposobów można ułożyć rzad o długości n za pomocą k nierozróżnialnych kości domina (prostokąty o wymiarach 2×1) i odpowiednio liczby nierozróżnialnych kwadratów 1×1 ?*

Przykładowe ułożenia dla trzech kości domina ($k = 3$) i długości rzędu $n = 9$ przedstawia rys. 2.10. Oczywiście, aby łączna długość rzędu wynosiła n , kwadratów musi być $n - 2k$. Liczbę różnych ułożen oznamy jako d_k^n . Kwadratów (w częstym formułowaniu zadań z dominem kwadraty określa się mianem *puścinych miejsc*) jest zawsze $n - 2k$, mamy też elementarne ograniczenie $2k \leq n$ – długość samych domin nie może przekraczać całej długości rzędu. Rozwiążanie problemu (2.1) jest bardzo proste, jeśli uzmysłowimy sobie, że mamy do czynienia z permutacjami z powtórzeniami. Mamy dwa rodzaje obiektów, kości domina jest $N_1 = k$, kwadratów jest $N_2 = n - 2k$, łącznie mamy $N = N_1 + N_2 = n - k$ obiektów, więc zgodnie z twierdzeniem (2.4) istnieje

$$d_k^n = \frac{N!}{N_1!N_2!} = \frac{(n-k)!}{k!(n-2k)!} = \binom{n-k}{k} \quad (2.29)$$

możliwości. Dla $n = 9$ i $k = 3$ daje to pokaźną liczbę $\binom{6}{3} = 20$.



Rysunek 2.10: Przykładowe liniowe pokrycia odcinka o długości 9 przy użyciu trzech kości domina, co pozostawia 3 puste miejsca

A ile jest wszystkich pokryć długości n , oznaczonych jako a_n ? Musimy po prostu zsumować po wszystkich możliwych liczbach kości domina k , zatem

$$a_n = \sum_{k=0}^{\lfloor n/2 \rfloor} d_k^n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k}, \quad (2.30)$$

gdzie funkcja podłogi (1.15) w górnej granicy sumowania wynika z ograniczenia $2k \leq n$. Zostawmy na moment przeprowadzenie sumowania i spróbujmy otrzymać wynik w inny sposób. Wyobraźmy sobie rząd długości n . Rząd ten na początku może mieć (1) kwadrat, a za nim rząd długości $n-1$, lub też (2) kość domina, a za nią rząd długości $n-2$ (zob. rys. 2.11). A zatem liczba wszystkich ułożen w rzędzie o długości n jest sumą ułożen z sytuacji (1) i (2). Znajdujemy więc rekurencję

$$a_n = a_{n-1} + a_{n-2}. \quad (2.31)$$

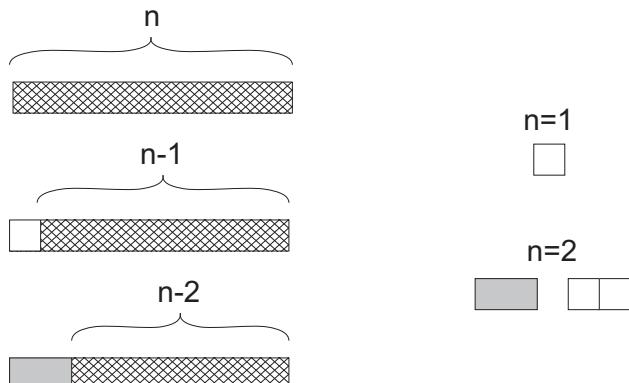
Ponadto (zob. rys. 2.11)

$$a_1 = 1, \quad a_2 = 2. \quad (2.32)$$

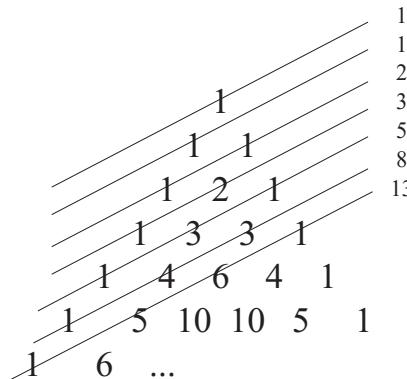
Jest to nic innego jak przepis na ciąg Fibonacciego! Liczba wszystkich pokryć o długości n to liczba Fibonacciego F_{n+1} , gdzie przesunięcie wskaźnika wynika z „przesuniętych” w stosunku do rekurencji Fibonacciego warunków początkowych: $a_1 = 1 = F_2$, $a_2 = 2 = F_3$. Wracając do sumy (2.30), widzimy, że

$$\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k} = F_{n+1},$$

co oznacza, że w trójkącie Pascala ukryte są też liczby Fibonacciego. Pokazane jest to na rys. 2.12: sumowanie wzdłuż zaznaczonych ukośnych linii odpowiada



Rysunek 2.11: Lewa strona: ilustracja rekurencji (2.31). Każde ułożenie o długości n może na początku mieć kwadrat, a za nim rząd długości $n-1$, lub kość domina, a za nią rząd długości $n-2$. Prawa strona: wszystkie możliwe ułożenia dla $n = 1$ i $n = 2$, dające warunki początkowe (2.32)



Rysunek 2.12: Sumowanie liczb w trójkącie Pascala wzdłuż zaznaczonych linii daje ciąg Fibonacciego

sumie (2.33), np.

$$\binom{4}{0} + \binom{3}{1} + \binom{2}{2} = 1 + 3 + 1 = 5 = a_4.$$

Argument oparty na rys. 2.11 można też wykorzystać dla wyprowadzenia rekurencji dla samych liczb d_k^n . Otrzymujemy

$$d_k^n = d_k^{n-1} + d_{k-1}^{n-2}, \quad 0 \leq k \leq \lfloor n/2 \rfloor, \quad (2.33)$$

z odpowiednimi warunkami początkowymi $d_0^1 = d_0^2 = d_1^2 = 1$. Obniżenie indeksu do $k-1$ w d_{k-1}^{n-2} wynika z faktu, że w tym przypadku ujmujemy jedną kość domina, patrz trzeci rzząd na lewej części rys. 2.11. Za pomocą tej rekurencji możemy szybko zbudować analogiczną do trójkąta Pascala tabelę 2.1 „liczb dominowych” d_k^n .

Tabela 2.1: Liczby dominowe d_k^n , czyli liczba sposobów ułożenia w rzędzie k kości domina (prostokąty 2×1) i $n - 2k$ kwadratów 1×1 . Rząd ma długość n . Konstrukcja tabeli przebiega zgodnie z rekurencją (2.33), czyli każdy wyraz jest sumą wyrazu w wierszu wyżej bezpośrednio nad nim oraz wyrazu dwa rzędy wyżej i o jeden w lewo. Przykładowo $15 = 10 + 5$ zaznaczono ramkami

	$k = 0$	1	2	3	4
$n=1$	1				
2	1	1			
3	1	2			
4	1	3	1		
5	1	4	3		
6	1	5	6	1	
7	1	6	10	4	
8	1	7	15	10	1
9	1	8	21	20	5

Kolejny problem jest bardzo podobny, tym razem jednak układamy domina i kwadraty wzduż okręgu.

Def. 2.13. [pokrycie okręgu dominem i kwadratami] Na ile sposobów można pokryć n ponumerowanych punktów na okręgu za pomocą k nierozróżnialnych kostek domina i odpowiednio liczby nierozróżnialnych kwadratów? Kość domina pokrywa dwa punkty, a kwadrat jeden.

Przykład przedstawiony jest na rys. 2.13.

Z rys. 2.14 wynika natychmiast, że liczba wszystkich pokryć okręgu o długości n wynosi

$$b_n = a_{n-1} + 2a_{n-2},$$

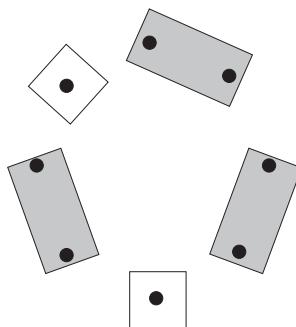
gdzie $a_n = F_{n+1}$ są liczbami pokryć odcinka.

Oznaczmy teraz liczbę różnych pokryć okręgu o długości n za pomocą k kości domina jako c_k^n . Liczby te można łatwo wyrazić posługując się liczbami d_k^n – rozumowanie oparte jest o rys. 2.14. Każde pokrycie n punktów na okręgu zawiera jedną z trzech sytuacji. Na rysunku po lewej punkt 1 pokryty jest kwadratem, zatem pozostałe $n-1$ punktów pokrytych jest liniowo (powyższy problem 2.1) za pomocą k kości domina. W dwóch pozostałych sytuacjach ze środkowej i prawej strony rysunku punkt 1 pokryty jest kościami domina, a więc pozostałe $n-2$ punktów pokrytych jest liniowo $k-1$ kościami domina. Zauważmy, że te dwie sytuacje są różne, rozróżniamy bowiem punkty na okręgu. W związku z tym dostajemy równość

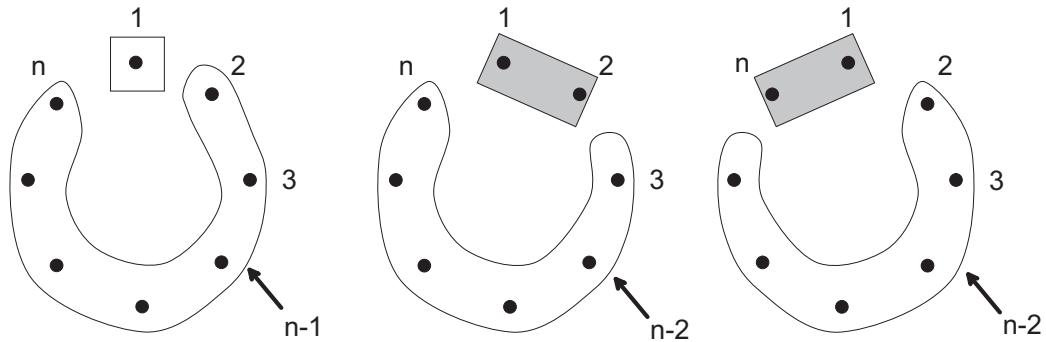
$$c_k^n = d_k^{n-1} + 2d_{k-1}^{n-2} = \binom{n-k-1}{k} + 2 \binom{n-k-1}{k-1} = \frac{n}{n-k} \binom{n-k}{k}, \quad (2.34)$$

gdzie ostatnia równość wynika z elementarnych przekształceń.

W powyższym rozumowaniu jest jedna nieścisłość. Otóż dla $n=2$ i $k=1$ mamy sytuację szczególną, nie mamy bowiem jeszcze okręgu, zatem nie możemy



Rysunek 2.13: Przykładowe pokrycie $n=8$ punktów na okręgu $k=3$ kościami domina i dwoma kwadratami



Rysunek 2.14: Pokrycie n punktów na okręgu za pomocą k kości domina (oraz $n - 2k$ kwadratów). Każde pokrycie zawiera jedną z trzech przedstawionych sytuacji. Na rysunku po lewej punkt 1 pokryty jest kwadratem, zatem pozostałe $n - 1$ punktów pokrytych jest liniowo za pomocą k kości domina. W dwóch sytuacjach ze środkowej i prawej strony rysunku punkt 1 pokryty jest kością domina, a więc pozostałe $n - 2$ punktów pokrytych jest liniowo $k - 1$ kości domina. Prowadzi to natychmiast do wyniku (2.34)

odróznić dwóch ułożeniu kości domina ze środkowej i prawej części rys. 2.14. Po prostu mamy tylko dwa punkty i jest jedna możliwość pokrycia ich dominem. W związku z tym

$$c_1^2 = 1,$$

a nie 2, jak wynikałoby naiwnie ze wzoru (2.34). Poprawny zapis wyniku jest następujący:

$$c_k^n = \frac{n}{n-k} \binom{n-k}{k}, \quad n \geq 3, \quad (2.35)$$

$$c_0^1 = 1, \quad c_0^2 = 1, \quad c_1^2 = 1. \quad (2.36)$$

Ćwiczenia

- 2.1. Udowodnij wzór (2.3).
 - 2.2. Używając inteligentnej sztuczki polegającej na zapisaniu
- $$n!^2 = (1 \cdot 2 \cdot 3 \dots n)(n \dots 3 \cdot 2 \cdot 1) = 1 \cdot n \cdot 2 \cdot (n-1) \dots n \cdot 1,$$
- uzyskaj dolne i górne ograniczenia dla $n!$.
- 2.3. Wyprowadź analog wzoru Stirlinga dla $n!!$.
 - 2.4. Ile k -literowych słów można zapisać za pomocą n liter alfabetu w taki sposób, aby (a) sąsiednie litery były różne, (b) pierwsza i ostatnia litera były różne?
 - 2.5. Rozważ zbiór A o n elementach, z którego wydzielono dwa podzbiory rozłączne: pierwszy X i drugi Y . Na ile sposobów można to zrobić?

- 2.6. Rozważ wielokąt wypukły o wierzchołkach w przypadkowych położeniach. Ile punktów tworzą przecięcia przekątnych tego wielokąta? (Na przykład dla czworokąta dwie przekątne przecinają się w jednym punkcie.)
- 2.7. Ile jest sposobów rozmieszczenia n nieroróżniczalnych obiektów w k rozróżnialnych pudłach, przy czym pudła mogą pozostać puste?
- 2.8. Wybór „ze zwracaniem” polega na tym, że po wybraniu obiektu i odnotowaniu wyniku obiekt zwracamy do puli, aby mógł być wybrany ponownie. Na ile sposobów można wybrać ze zwracaniem k obiektów spośród n rozróżnialnych obiektów, jeśli kolejność wyboru nie jest istotna? Jest to tzw. kombinacja z powtórzeniami, \bar{C}_k^n .
- 2.9. Na ile sposobów sześć osób może wybrać po kawałku ciasta z patery, na której są po dwa kawałki ciasta z pięciu różnych rodzajów?
- 2.10. Korzystając z funkcji tworzącej, pokaż, że dla $n > 1$

$$\sum_{k=0}^n k(-1)^k \binom{n}{k} = 0.$$

Sprawdź wynik dla kilku pierwszych wartości n , wykorzystując trójkąt Paszcal'a.

- 2.11. Na ile sposobów można wejść na wieżę po n schodach, robiąc kroki po jednym lub po 2 schody?
- 2.12. Udowodnij na podstawie zasady szufladkowej Dirichleta, że na dowolnie pomalowanej dwoma kolorami płaszczyźnie można znaleźć prostokąt o wierzchołkach tego samego koloru.
- 2.13. Wykaż, że w grupie $n \geq 2$ osób przynajmniej dwie mają po tyle samo znajomych (zakładamy, że relacja znajomości jest zwrotna, tzn. jeśli osoba A zna osobę B, to osoba B zna osobę A).
- 2.14. Wyprowadź wzory na

$$\sum_{k=0}^n k \binom{n}{k} \text{ oraz } \sum_{k=0}^n k^2 \binom{n}{k}.$$

- 2.15. Wyprowadź wzory na liczbę pokryć odcinka oraz okręgu o długości n za pomocą k sztuk domina o długości 3 i kwadratami o długości 1.
- 2.16. Ile jest wszystkich pokryć odcinka o długości n za pomocą kości domina o długości 2 i kwadratów o długości 1, gdzie domina i kwadraty występują w dwóch kolorach?
- 2.17. Ile jest wszystkich pokryć prostokąta o wymiarach $3 \times n$ klepkami parkietu o wymiarach 2×1 ?

Rozdział 3

Elementy rachunku prawdopodobieństwa

Zliczanie przedstawione w rozdz. 2 i kolejnym rozdz. 4 w praktyce służy bardzo często obliczaniu *prawdopodobieństwa* zdarzeń losowych. W tym rozdziale przypomnimy więc (poznane jeszcze w szkole) podstawowe wiadomości dotyczące rachunku prawdopodobieństwa, a jako nietrywialne zastosowanie wyjaśnimy *paradoks urodzin*. Poznamy też bardzo użyteczną i często stosowaną metodę zliczania, tzw. regułę włączeń i wyłączeń. Rozwiążemy jako ilustracje znane problemy dotyczące kibiców Korony Kielce (!) oraz gości na przyjęciu. Przypomnimy też pojęcie *prawdopodobieństwa warunkowego* i związane z nim Tw. Bayesa, które zobrazujemy *paradoksem hipochondryka*.

3.1 Prawdopodobieństwo dyskretne

Klasyczna definicja prawdopodobieństwa pochodzi od Laplace'a¹. Określamy przestrzeń zdarzeń elementarnych $\Omega = \{\omega\}$ – jest to zbiór wszystkich zdarzeń ω , które mogą zajść w danym problemie, przy czym zdarzenia te są od siebie *niezależne* i są *jednakowo możliwe*. Rozważmy podzbiór $A \subset \Omega$, który nazywamy zdarzeniem (zauważmy, że nie jest to samo co zdarzenie elementarne, które jest elementem zbioru, a nie podzbiorem).

Def. 3.1 (prawdopodobieństwo klasyczne). *Prawdopodobieństwem zajścia zdarzenia A nazywamy iloraz*

$$P(A) = \frac{|A|}{|\Omega|},$$

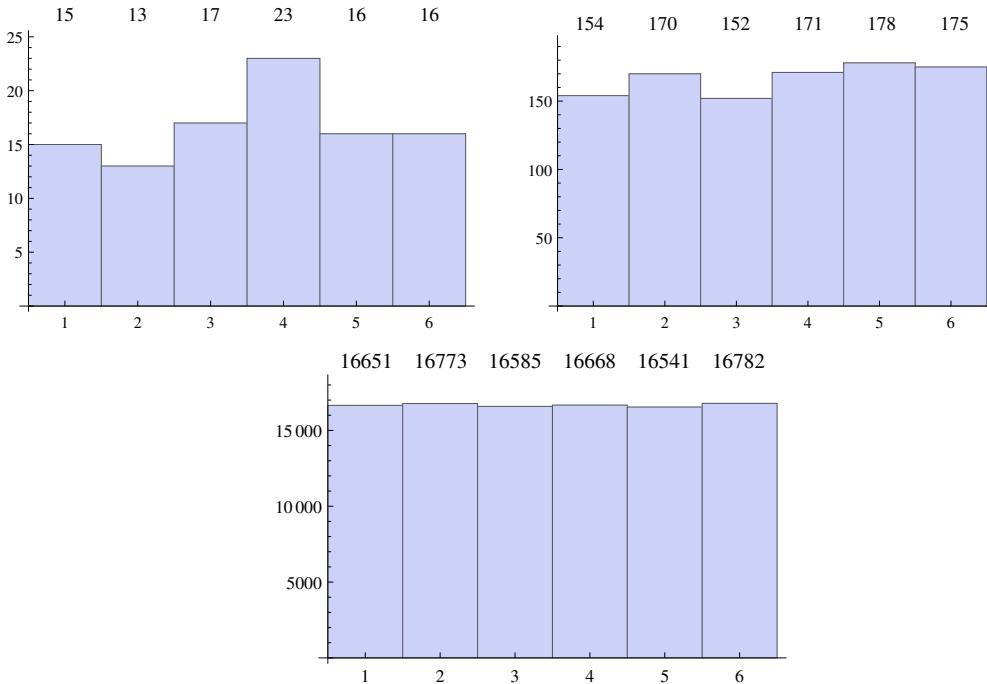
gdzie $|\cdot|$ oznacza liczbę elementów (tj. moc) zbioru.

Definicja ta oznacza, że aby określić prawdopodobieństwo, musimy po prostu policzyć elementy zbiorów Ω i A . Często jest to jednak skomplikowane, ponieważ zbiór A może być określony w taki sposób, że liczenie jest trudne. Tym właśnie zajmuje się kombinatoryka.

Przykład 3.1. Wyświechtanym przykładem, ale dlatego, że najlepszym, jest gra w kości. Powiedzmy, że rzucamy jeden raz kością. Wtedy przestrzeń zdarzeń elementarnych to zbiór $\Omega = \{\square, \blacksquare, \blacksquare\square, \square\square, \blacksquare\blacksquare, \blacksquare\square\square\}$, czyli zbiór wszystkich zdarzeń, które mogą zajść. Powiedzmy, że interesuje nas wyrzucenie $\blacksquare\square$, czyli $A = \{\blacksquare\square\}$. Zliczamy i dostajemy $P(\blacksquare\square) = |A|/|\Omega| = 1/6$. Innymi zdarzeniami są np. $\{\square, \blacksquare\}$, tj. wyrzucenie jedynki lub dwójki, czy też $\{\square, \blacksquare, \blacksquare\square\}$ – wyrzucenie dowolnej liczby nieparzystej. Ich prawdopodobieństwa wynoszą, odpowiednio, $1/3$ i $1/2$.

Ukrytym założeniem jest tu symetryczność kostki, potrzebna, aby wszystkie zdarzenia elementarne były jednakowo prawdopodobne. Jak sprawdzić, czy ktoś nie sfałszował kostki, np. nie włożył do środka ołowianego ciężarka, przesuwając środek ciężkości w kierunku \square ? Istnieje prosty ale uciążliwy test: musimy rzucić kostką bardzo, bardzo wiele razy i policzyć, ile razy wypadły poszczególne oczka. Rysunek 3.1 przedstawia wynik takiego testu. Oczywiście kostką rzucał tu za nas komputer, a rolę kostki pełnił generator liczb losowych. Na rysunku mamy kolejno wynik rzucania 100 razy, 1000 razy i 100000 razy. Nad słupkami histogramów zaznaczono sumaryczną liczbę rzutów, dla których wypadły poszczególne oczka. Oznaczmy te liczby (krotności) jako N_i , gdzie i oznacza liczbę oczek. I tak dla 100 rzutów (lewa góra część rys. 3.1) $N_{\square} = 15$, $N_{\blacksquare} = 13$, $N_{\blacksquare\square} = 17$, $N_{\square\square} = 23$, $N_{\blacksquare\blacksquare} = 16$, $N_{\blacksquare\square\square} = 16$ – widzimy, że różne oczka wcale nie wypadły po tyle samo razy! Mówimy, że ich krotności *fluktują*. Gdy powtórzymy próbę, wykonując

¹ Pierre-Simon de Laplace (1749-1827), francuski matematyk, astronom i fizyk, jeden z ojców rachunku prawdopodobieństwa, przyczynił się znacznie do rozwoju analizy matematycznej.



Rysunek 3.1: Histogramy rzutów kostką dla 100, 1000 i 100000 prób. Poszczególne słupki odpowiadają danej liczbie oczek, a uzyskane krotności podano nad słupkami

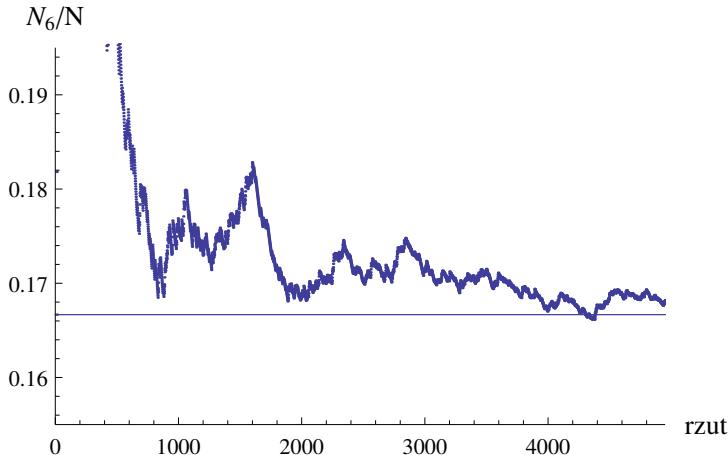
1000 rzutów (prawy górny wykres), to sytuacja wygląda „lepiej”, czyli wysokość słupków są sobie *względnie* bliższe. Teraz kolejne N_i wynoszą 154, 170, 152, 171, 178, 175. Chociaż te liczby są różne, to ich *względna* różnica jest znacznie mniejsza niż dla 100 rzutów. Dla 100000 rzutów względna wysokość słupków jest wzajemnie jeszcze bliższa, co ujawnia dolna część rys. 3.1. Można sobie łatwo wyobrazić, że powtarzając doświadczenie nieskończonie wiele razy, uzyskamy stosunki N_i/N_j zmierzające do jedności.

Taka procedura wielokrotnych losowań sprowadza nas do tzw. *częstościowej* definicji prawdopodobieństwa. Możemy określić $P(i)$ za pomocą przejścia granicznego

$$P(i) = \lim_{N \rightarrow \infty} \frac{N_i}{N}, \quad (3.1)$$

(tego typu podejście zaproponował von Mises²), gdzie N jest liczbą wszystkich rzutów w próbie. Związek (3.1) jest przejawem tzw. *prawa wielkich liczb*. Dla $N = 100000$ jesteśmy istotnie niedaleko granicy, gdyż nasz eksperyment (dolina rys. 3.1) daje dla wartości N_i/N , $i = 1, \dots, 6$, kolejno 0.1665, 0.1677, 0.1659, 0.1667, 0.1654, 0.1678, a więc bardzo blisko $1/6$. Jeśli chcemy być jeszcze bliżej, musimy rzucić jeszcze więcej razy. Podobna analiza przedstawiona jest

² Richard von Mises (1883-1953), austriacki matematyk, który przyczynił się do rozwoju teorii prawdopodobieństwa i statystyki, a także hydrodynamiki i aerodynamiki.



Rysunek 3.2: Wynik eksperymentu rzucania kostką 5000 razy. Na osi poziomej oznaczono numer rzutu N , a na pionowej łączną liczbę uzyskanych do danego rzutu szóstek podzieloną przez liczbę wykonanych rzutów, co daje aktualną średnią krotność szóstek N_6/N . Dla dużych N średnia ta zmierza do wartości $1/6$, zaznaczonej poziomą kreską

na rys. 3.2, który przedstawia wynik eksperymentu rzucania kostką $N = 5000$ razy. Po każdym kolejnym rzucie dzielimy liczbę uzyskanych dotychczas \blacksquare przez bieżącą liczbę wykonanych rzutów. Daje nam to aktualną średnią krotność \blacksquare . Widzimy, że dla bardzo dużych N ta średnia mazolnie zmierza, fluktuując, do wartości $1/6$, zaznaczonej poziomą kreską. Zachowanie to jest zgodne z (3.1). Można pokazać, że odstępstwo aktualnej średniej krotności od wartości $1/6$ jest rzędu $1/\sqrt{N}$, więc granica osiągana jest stosunkowo wolno.

W praktyce nie sprawdzamy jednak kupionych kostek do gry poprzez wieleokrotne rzucanie. Zakładamy, że są symetryczne. Jest to zatem *aksjomat!* Czyńmy tak na podstawie oględzin kostki, wcześniejszych doświadczeń, zaufania do sprzedawcy itd. Ponadto rzucamy za pomocą kubeczka, żeby nie widzieć kostki podczas rzutu i nie ulegać pokusie wspomożenia losu! Przyjmujemy, że $P(i)/P(j) = 1$, czyli $P(i) = 1/6$ dla każdego i . Jeszcze zanim zaczniemy rzucić, wiemy, na mocy prawa (3.1), że $\lim_{N \rightarrow \infty} \frac{N_i}{N} = 1/6$, zatem oczekujemy, że przy dostatecznie dużej liczbie rzutów dostaniemy średnio co szósty raz \blacksquare .

W def. 3.1 istotna też jest niezależność zdarzeń elementarnych. W przypadku rzutów kostką jest ona oczywista. Wynik kolejnego rzutu nie zależy od poprzedniego, nie mamy żadnych *korelacji* (tj. wpływu jednego zdarzenia na drugie), więc zdarzenia $\square, \blacksquare, \blacksquare\blacksquare, \blacksquare\square, \blacksquare\blacksquare\square, \blacksquare\blacksquare\blacksquare$ są *niezależne*.

Na koniec abstrakcyjna i matematycznie najbardziej ogólna definicja *aksjomatyczna* prawdopodobieństwa Kołmogorowa³, oparta na tzw. pojęciu miary

³ Andriej Nikołajewicz Kołmogorow (1903-1987), radziecki matematyk, który bardzo znacząco przyczynił się do rozwoju teorii prawdopodobieństwa, topologii, mechaniki klasycznej, teorii turbulencji i złożoności algorytmów.

(definicja ta jest niezbędna, gdy mamy do czynienia z przestrzenią zdarzeń, która stanowi zbiór ciągły).

Def. 3.2 (aksjomatyczna definicja prawdopodobieństwa). *Rozważmy rodzinę podzbiorów M zbioru Ω . Prawdopodobieństwem nazywamy funkcję $P : M \rightarrow R$ o własnościach*

1. $\forall A \in \Omega : P(A) \geq 0$,
2. $P(\Omega) = 1$,
3. Dla dowolnych parami rozłącznych zbiorów A_i zachodzi

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i).$$

Pierwszy warunek mówi, że prawdopodobieństwo zdarzenia jest nieujemne. Drugi oznacza, że prawdopodobieństwo zajścia jakiegokolwiek zdarzenia, tzn. *zdarzenia pewnego*, wynosi 1. Trzeci warunek mówi, że prawdopodobieństwo zdarzeń rozłącznych jest równe sumie prawdopodobieństw, nawet dla nieskończonej sumy. Znak \cup oznacza tu sumę teoriomnogościową, tzn. $\bigcup_{i=1}^{\infty} A_i = A_1 \cup A_2 \cup A_3 \cup \dots$, a określenie *parami rozłączne* oznacza, że każda para jest rozłączna. Dla dwóch zbiorów rozłącznych warunek 3 mówi po prostu, że $P(A_1 \cup A_2) = P(A_1) + P(A_2)$. Zauważmy, że dla przypadku zbiorów skończonych definicja aksjomatyczna redukuje się do def. 3.1. Z definicji wynika natychmiast kilka twierdzeń:

Tw. 3.1. $P(\emptyset) = 0$

Dowód: $A \cup \emptyset = A$, $A \cap \emptyset = \emptyset$, zatem $P(A) = P(A \cup \emptyset) = P(A) + P(\emptyset)$, skąd $P(\emptyset) = 0$. \square

Tw. 3.2. $A \subset B \Rightarrow P(A) \leq P(B)$.

Dowód: Mamy $B = A \cup (B \setminus A)$ ⁴, przy czym A i $B \setminus A$ są rozłączne, wobec czego $P(B) = P(A) + P(B \setminus A) \geq P(A)$. \square

Wniosek 3.1. Biorąc $B = \Omega$, mamy $P(A) \leq 1$.

Tw. 3.3. Prawdopodobieństwo zdarzenia przeciwnego (dopełniającego) $\bar{A} = \Omega \setminus A$ wynosi

$$P(\bar{A}) = 1 - P(A).$$

⁴ Symbol \setminus oznacza tu różnicę teoriomnogościową zbiorów, w szczególności $B \setminus A = \{x : x \in B \wedge x \notin A\}$.

Dowód: Ponieważ A i \bar{A} są rozłączne, mamy $P(A) + P(\bar{A}) = P(\Omega) = 1$. \square

Przedyskutujemy teraz niezależność zdarzeń losowych.

Def. 3.3. Zdarzenia A i B nazywamy niezależnymi, jeśli $P(A \cap B) = P(A)P(B)$.

Przykład 3.2. Rozważmy rzut dwiema kostkami. Mamy możliwych $6 \cdot 6 = 36$ układów oczek, $|\Omega| = 36$:

$$\begin{aligned}\Omega = \{ & \square\square, \square\bullet, \bullet\square, \bullet\bullet, \square\square, \square\bullet, \\ & \bullet\square, \square\square, \square\bullet, \bullet\square, \square\square, \square\bullet \} \end{aligned}$$

Niech A oznacza „na pierwszej kostce wypadła \square , a na drugiej cokolwiek” oraz B oznacza „na drugiej kostce wypadła \square , a na pierwszej cokolwiek”. Czyli

$$\begin{aligned}A = \{ & \square\square, \square\bullet, \bullet\square, \bullet\bullet, \square\square, \square\bullet \}, \\ B = \{ & \bullet\square, \square\square, \square\bullet, \bullet\square, \square\square, \square\bullet \},\end{aligned}\tag{3.2}$$

oraz

$$A \cap B = \{\square\square\}.$$

Oczywiste zliczanie daje $P(A) = P(B) = 6/36 = 1/6$, $P(A \cap B) = 1/36$, wobec czego $P(A \cap B) = P(A)P(B)$ i zdarzenia A i B , zgodnie z naszą intuicją, są niezależne. Wyrzucenie \square na jednej kostce w żaden sposób nie wpływa na drugą kostkę.

Przykład 3.3. Wyobraźmy sobie teraz, że ktoś skleił dwie kostki ściankami \square i \bullet w taki sposób, że możliwe zdarzenia to (zaniedbujemy możliwość, że sklejone kostki staną „na sztorc”)

$$\begin{aligned}\Omega = \{ & \square\square, \square\bullet, \bullet\square, \bullet\bullet \}, \\ A = \{ & \bullet\bullet \}, \\ B = \{ & \square\bullet \}, \\ A \cap B = \{ & \bullet\bullet \}.\end{aligned}$$

W tym przypadku $P(A \cap B) = 1/4$, natomiast $P(A)P(B) = 1/4 \cdot 1/4 = 1/16$ i zdarzenia nie są niezależne, co dla sklejonych kostek jest oczywiste! Wyrzucenie \square na jednej oznacza również uzyskanie \bullet na drugiej.

3.2 Paradoks urodzin

Bardzo ładny problem obrazujący technikę zliczania to tzw. *paradoks urodzin*. Zazwyczaj wprowadza się go w nieco teatralny sposób. Wykładowca rozpoczynając zajęcia z kilkudziesięcioosobową osobową grupą studentów, mówi: „Mam dziwne przeczucie, że w tej sali są osoby, które mają urodziny tego samego dnia!” i praktycznie zawsze ma rację: „Co za przypadek!”. Słowo *paradoks* w nazwie problemu bierze się stąd, że intuicja (przynajmniej kombinatorycznie niewyrobiona intuicja) podpowiada nam, że liczba ta jest niewielka. Policzymy bowiem prawdopodobieństwo P_w zdarzenia, że przynajmniej jeden z 60 studentów ma urodziny wtedy, kiedy wykładowca, tj. 17 lipca. Łatwiej policzyć prawdopodobieństwo zdarzenia dopełniającego, czyli że nikt nie ma urodzin tego dnia, co wynosi $(364/365)^{60} \simeq 0.85$. A zatem

$$P_w = 1 - (364/365)^{60} \simeq 0.15.$$

Jest to niewielka szansa, więc wykładowca, aby nie stracić autorytetu, nie powie „Mam przeczucie, że ktoś z Państwa ma urodziny wtedy, co ja”. Na podstawie małej wartości P_w powiedzielibyśmy więc być może, że wynik dla paradoksu urodzin jest podobny, no może 50%. W rzeczywistości prawdopodobieństwo, że w grupie 60 osób co najmniej dwie mają urodziny tego samego dnia wynosi aż 99.4%!

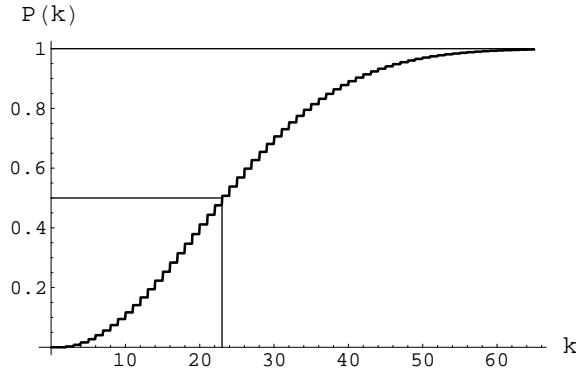
Kombinatoryczna definicja problemu jest następująca:

Problem 3.1 (paradoks urodzin). *Z n rozróżnialnych obiektów wybieramy ze zwracaniem k obiektów. Jakie jest prawdopodobieństwo, że m lub więcej wybranych obiektów jest identycznych?*

Obiektami jest oczywiście $n = 365$ dni roku, które przydzielone są k studentom, a $m = 2$. Zakładamy też, że daty urodzin są rozłożone równomiernie, tj. prawdopodobieństwo, że dana osoba urodziła się w danym dniu wynosi $1/365$, oraz że daty urodzin różnych osób są niezależne (nie ma bliźniąt, trojaczków). Założenie, że urodziny rozłożone są równomiernie nie jest do końca słuszne, bo podlegają one sezonowym wahaniom oraz perturbacjom wywołanym np. awariami sieci energetycznych⁵.

Znowu, jak często bywa, znacznie łatwiej jest policzyć prawdopodobieństwo \bar{P} zdarzenia dopełniającego, tzn. zdarzenia, że nikt z k osób nie ma urodzin tego samego dnia. Wyobraźmy sobie, że przydzielamy daty urodzin. Wszystkich możliwości jest n^k (wariacje k -elementowe n -elementowego zbioru), więc $|\Omega| = n^k = 365^{60}$. W zdarzeniu \bar{P} pierwszej osobie możemy dać dowolną datę, mamy więc $n = 365$ możliwości, dla drugiej osoby mamy już tylko $n - 1 = 364$ możliwości, bo jedna data jest już zajęta, dla trzeciej osoby mamy $n - 2 = 363$

⁵ Istnieje mit głoszący, że awaria prądu w Nowym Jorku 14 sierpnia 2003 r. spowodowała istny boom urodzin 9 miesięcy później! Ponoć statystki tego jednak nie potwierdzają...



Rysunek 3.3: Paradoks urodzin: Prawdopodobieństwo $P(k)$, że co najmniej dwie osoby w grupie k osób mają urodziny w tym samym dniu. $P(23)$ nieznacznie przekracza wartość $1/2$, $P(60)$ jest już praktycznie równe 1

możliwości itd. Dostajemy więc ogólnie następujący wzór:

$$P = 1 - \bar{P} = 1 - \frac{n(n-1)\dots(n-k+1)}{n^k} = 1 - \frac{n!}{(n-k)! n^k}. \quad (3.3)$$

Rysunek 3.3 pokazuje zależność P od k . Zauważmy, że już $P(23) = 0.507$, a więc wystarczy, by na wykład przyszły tylko 23 osoby, aby wykładowca miał większą szansę znalezienia niż nieznalezienia dwóch osób o tym samym dniu urodzin. Natomiast $P(60) = 0.994$, już bardzo blisko 1, $P(90) = 0.9999994$, $P(120) = 0.9999999998$. Oczywiście, dla $k > 365$ mamy $P(k) = 1$, bo na mocy zasadyszufladkowej nie jest możliwe, by więcej niż 365 osób miało różne daty urodzin (w naszej analizie pomijamy nieszczęśników urodzonych 29 lutego!). Zauważmy, że $P(60)$ jest dużo większe od prawdopodobieństwa P_w policzonego na początku podrozdziału, gdzie wyróżniliśmy datę urodzin wykładowcy. Rozwiązańeparadoksu tkwi właśnie w fakcie, że przy obliczaniu P_w ustaliliśmy konkretną datę, podczas gdy w problemie urodzin interesuje nas tylko, czy osoby mają ten sam dzień urodzin, ale nie specyfikujemy daty, która może być dowolnym z 365 dni roku. Powoduje to „wzrost kombinatoryki” i wyjaśniony powyżej efekt.

Za pomocą wzoru Stirlinga (2.7) możemy dla dużych n i k napisać przybliżoną postać wzoru (3.3):

$$1 - P \simeq \frac{\sqrt{2\pi n} e^{-n} n^n}{\sqrt{2\pi(n-k)} e^{-(n-k)} (n-k)^{n-k} n^k} = e^{-k} \left(1 - \frac{k}{n}\right)^{k-n-\frac{1}{2}}, \quad (3.4)$$

zatem

$$\log(1 - P) \simeq -k + \left(k - n - \frac{1}{2}\right) \log\left(1 - \frac{k}{n}\right). \quad (3.5)$$

Przyjmując, że $k \ll n$, możemy rozwinać logarytm, $\log(1-x) = -x - x^2/2 + \dots$, dzięki czemu dostajemy równanie

$$\log(1-P) = -k - (k-n - \frac{1}{2}) \left(\frac{k}{n} + \frac{k^2}{2n^2} \right) \simeq -\frac{k^2}{2n}, \quad (3.6)$$

gdzie zaniedbaliśmy człony rzędu k/n . W wyniku otrzymujemy

$$k \simeq \sqrt{-2 \log(1-P)n}. \quad (3.7)$$

Dla $P = 1/2$ mamy $k \simeq 1.17\sqrt{n}$, co przy $n = 365$ daje $k = 22.5$, z kolei dla $P = 7/8$ dostajemy $k \simeq 2.04\sqrt{n}$, czyli $k = 38.96$ dla $n = 365$. Dokładny rachunek daje odpowiednio $k = 23$ i $k = 39$, nasze oszacowanie asymptotyczne działa więc bardzo dobrze dla przyjętych liczb, co uwidacznia praktyczne znaczenie i dokładność wzoru Stirlinga. Wzór (3.7) pozwala na bardzo szybkie oszacowanie prawdopodobieństwa w problemie urodzin: np. dla $P = 1/2$ wystarczy policzyć pierwiastek z n i powiększyć wynik o 17%.

Do tej pory rozważaliśmy problem, w którym *przynajmniej* dwie osoby mają urodziny w tym samym dniu. Pozostałe osoby nas nie interesowały: mogły mieć urodziny każda osobno lub też niektóre z nich też mogły mieć wspólne urodziny. Pokażemy teraz prostą technikę zliczania, dzięki której można obliczyć prawdopodobieństwo wystąpienia wspólnych urodzin u *dokładnie* jednej pary, dokładnie dwóch par, dokładnie jednej trójki osób itd., w ogólności dla dowolnej konfiguracji. Od tej pory określenie *para*, *trójka* itd. oznacza osoby urodzone tego samego dnia. Dla ustalenia uwagi rozważmy prawdopodobieństwo, że wśród 100 osób mamy dokładnie jedną trójkę i dwie pary. Zliczanie będzie dwustopniowe: najpierw obliczymy liczbę możliwych *rozkładów dat* dla danej konfiguracji, a następnie liczbę możliwych obsadzeń danego rozkładu dat przez konkretne osoby. W naszym przykładzie mamy jeden dzień w roku, w którym urodziły się trzy osoby, dwa dni w których urodziły się po dwie osoby, 93 dni w których przyszły na świat pojedynczo pozostałe osoby oraz 269 dni, w których nie urodził się nikt. Taką konfigurację możemy oznaczyć jako

$$(3)(22)(11\dots11)(00\dots00), \quad (3.8)$$

gdzie jedynek jest 93, a zer 269. Pod tym zapisem kryje się wiele możliwości wyboru konkretnych dat, np. dzień, w którym urodziła się trójka to 13 listopada lub 1 maja itd. Na ile sposobów możemy przypisać daty do konfiguracji (3.8)? Jest to rozważany w Tw. 2.4 przypadek permutacji z powtórzeniami o krotnościach 1, 2, 93 i 269, co możemy zrobić na

$$d = \frac{365!}{1! 2! 93! 269!} \quad (3.9)$$

sposobów. Elementami permutowanymi są tu cztery rodzaje dat: dzień, w którym urodziła się trójka (krotność jeden), dzień, w którym urodziła się para (krotność

2), dzień, w którym urodziła się jedna osoba (krotność 93) i dzień, w którym nikt się nie urodził (krotność 269).

A teraz drugi etap zliczania: do danego rozkładu dat przypisujemy konkretne osoby. Wyobraźmy sobie, że ustaviamy je „w kolejce” dokładnie pod rozkładem (3.8). Osób jest 100, więc możemy je ustawić na $100!$ sposobów, jednak dla pierwszych trzech osób nie odgrywa roli kolejność, więc musimy tę liczbę podzielić przez $3!$. Podobnie, dla kolejnych osób tworzących dwie pary dzielimy przez $2!^2$. Mamy więc

$$p = \frac{100!}{3! 2!^2} \quad (3.10)$$

możliwości przydzielenia osób do danego rozkładu dat. Łącznie mamy więc dp możliwości ułożenia osób zgodnie z rozkładem (3.8). Wszystkich możliwości „przydziału” dat 100 osobom jest oczywiście 365^{100} (wariacje z powtórzeniami), zatem ostatecznie szukane prawdopodobieństwo wynosi

$$P = \frac{pd}{365^{100}} = \frac{1}{365^{100}} \frac{365!}{1! 2! 93! 269!} \frac{100!}{3! 2!^2} \simeq 0.0001. \quad (3.11)$$

Rozważmy teraz ogólny przypadek n dni i k osób. Oznaczmy przez l_i krotność zdarzenia, że dokładnie i osób ma urodziny tego samego dnia. W poprzednim przykładzie $l_3 = 1$, $l_2 = 2$, $l_1 = 93$, $l_0 = 269$, a pozostałe l_i wynoszą zero. Oczywiście, suma iloczynów liczb i i ich krotności daje całkowitą liczbę osób, k :

$$kl_k + (k-1)l_{k-1} + \dots + 3l_3 + 2l_2 + 1l_1 + 0l_0 = k. \quad (3.12)$$

Ponadto, suma krotności daje liczbę dni w roku, n :

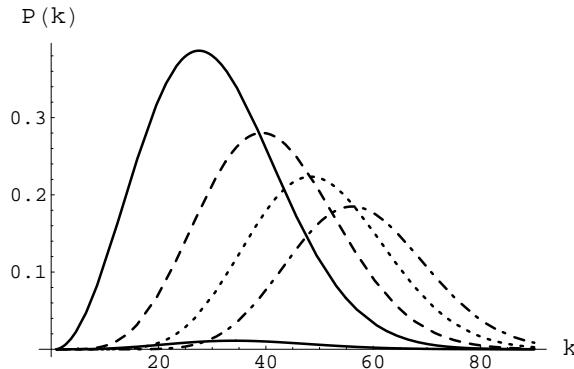
$$l_k + l_{k-1} + \dots + l_3 + l_2 + l_1 + l_0 = n. \quad (3.13)$$

Powtarzając powyższe rozumowanie, dostajemy natychmiast ogólne postaci wzorów (3.9-3.11):

$$\begin{aligned} d &= \frac{n!}{l_k! l_{k-1}! \dots l_2! l_1! l_0!}, \\ p &= \frac{k!}{k!^{l_k} (k-1)!^{l_{k-1}} \dots 2!^{l_2} 1!^{l_1} 0!^{l_0}}, \\ P &= \frac{1}{n^k} \frac{n!}{l_k! l_{k-1}! \dots l_2! l_1! l_0!} \frac{k!}{k!^{l_k} (k-1)!^{l_{k-1}} \dots 2!^{l_2}}. \end{aligned} \quad (3.14)$$

Możemy teraz podstawić do powyższego wzoru kilka najprostszych konfiguracji. Najpierw rozważmy prawdopodobieństwo, że dokładnie jedna para spośród k studentów ma wspólne urodziny. Mamy $l_2 = 1$, $l_1 = k-2$, $l_0 = 365 - 1 - (k-2)$, pozostałe l_i wynoszą 0, zatem wzór (3.14) daje

$$P = \frac{1}{365^k} \frac{365!}{(k-2)!(365-k+1)!} \frac{k!}{2!}. \quad (3.15)$$



Rysunek 3.4: Prawdopodobieństwa $P(k)$ uzyskania dokładnie jednej pary (linia ciągła), dwóch par (linia przerywana), trzech par, (linia kropkowana), czterech par (linia kreska-kropka) oraz trójki (linia ciągła tuż nad osią x) w funkcji liczby osób k

Dla $k = 23$ dostajemy $P = 0.363$, istotnie mniej niż wyprowadzone na początku rozdziału prawdopodobieństwo uzyskania co najmniej jednej pary, wynoszące 0.507. Podobnie, prawdopodobieństwo, że dokładnie trzy osoby mają urodziny w jednym dniu, $l_3 = 1$, $l_2 = 0$, $l_1 = k - 3$, $l_0 = 365 - k + 2$, wynosi

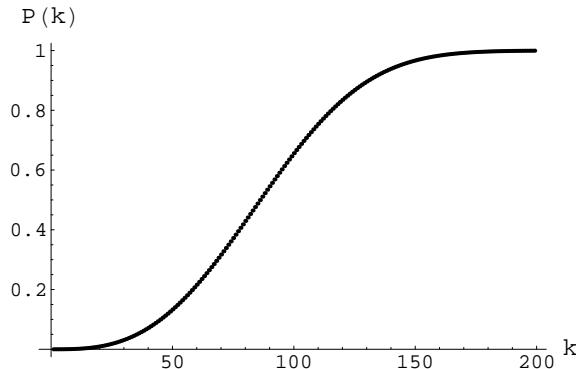
$$P = \frac{1}{365^k} \frac{365!}{(k-3)!(365-k+2)!} \frac{k!}{3!}, \quad (3.16)$$

co dla $k = 23$ daje zaledwie 0.007. W podobny sposób możemy otrzymać prawdopodobieństwo dowolnej konfiguracji. Rysunek 3.4 przedstawia zależność prawdopodobieństw otrzymywania kilku prostych konfiguracji w problemie urodzin. Zaauważmy, że prawdopodobieństwa te osiągają maksimum przy pewnym k , po czym maleją do zera przy rosnącym k . Na przykład prawdopodobieństwo, że dokładnie jedna para ma wspólne urodziny jest największe dla $k = 28$, gdzie wynosi 0.386. To malenie ma prostą interpretację. W miarę wzrostu k staje się coraz bardziej prawdopodobne, że oprócz naszej jednej pary pojawi się jeszcze inna para, trójka itd.

Zsumowane prawdopodobieństwa wszystkich konfiguracji dają oczywiście jeden. Liczba możliwych konfiguracji bardzo szybko rośnie z k i coraz więcej konfiguracji daje przyczynę do tej sumy. Dlatego, chociaż prawdopodobieństwa poszczególnych konfiguracji są dla dużych k niewielkie, ich suma wynosi jeden.

A jakie jest prawdopodobieństwo, że *co najmniej* trzy osoby mają urodziny tego samego dnia? Podobnie jak w oryginalnym paradoksie urodzin, interesuje nas teraz nie jedna konkretna konfiguracja, ale wiele konfiguracji, w których trzy lub więcej osób urodziło się tego samego dnia, a inne mogą mieć urodziny razem, osobno, w dowolnym układzie. Prawdopodobieństwo to można obliczyć, wykonując następujący rachunek:

$$\begin{aligned} P(\text{co najmniej jedna trójka}) &= P(\text{co najmniej jedna para}) - \\ &- P(\text{dokładnie jedna para}) - P(\text{dokładnie dwie pary}) - P(\text{dokładnie trzy pary}) - \\ &\dots \end{aligned}$$



Rysunek 3.5: Prawdopodobieństwo uzyskania co najmniej jednej trójki w problemie urodzin w funkcji liczby osób k . Wartość $1/2$ przekroczena jest dla $k \geq 88$

Od konfiguracji „co najmniej jedna para” odejmujemy konfiguracje składające się wyłącznie z samych par. W ten sposób zostają konfiguracje zawierające trójki, czwórki itd., określane przez nas jako „co najmniej jedna trójka”. Następnie (korzystając z komputera) stosujemy wzory (3.3) i (3.14). Wynik przedstawiony jest na rys. 3.5. Dla $k \geq 88$ prawdopodobieństwo przekracza wartość $1/2$. Trzeba więc większego gremium, aby wykładowca powiedział: „Mam przeczucie, że wśród Państwa są trzy osoby, które mają urodziny tego samego dnia!”.

3.3 Zasada włączania i wyłączania

Jak argumentowaliśmy w podrozdziale 3.1, definicja prawdopodobieństwa $P(A)$ sprowadza jego wyznaczenie do znalezienia mocy zbiorów A i Ω . Przy tym zliczaniu bardzo pomocna jest pewna prosta zasada, znana jako *zasada włączania i wyłączania*. Cała idea zasadza się na addytywności mocy zbiorów rozłącznych. Popatrzmy na lewy diagram rys. 3.6, przedstawiającego tzw. diagramy Venna (znane też jako diagramy Eulera). Możemy zapisać sumę zbiorów A i B jako sumę zbiorów rozłącznych, $A \cup B = (A \setminus B) \cup (A \cap B) \cup (B \setminus A)$, skąd liczby elementów zbiorów spełniają równość

$$|A \cup B| = |A \setminus B| + |A \cap B| + |B \setminus A|. \quad (3.17)$$

Z drugiej strony $A = (A \setminus B) \cup (A \cap B)$ oraz $B = (B \setminus A) \cup (A \cap B)$, zatem

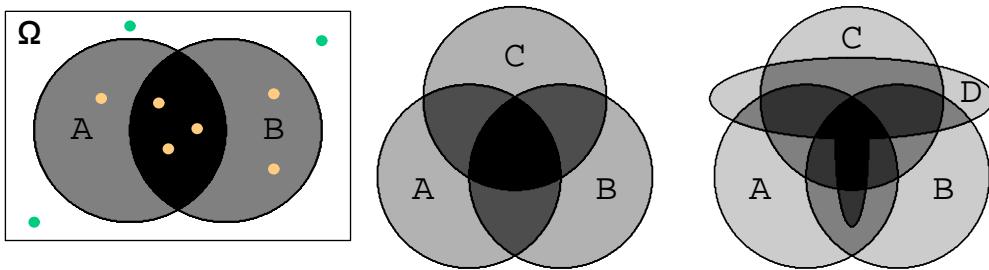
$$|A| = |A \setminus B| + |A \cap B|, \quad |B| = |B \setminus A| + |A \cap B|. \quad (3.18)$$

Łącząc wzory (3.17) i (3.18), otrzymujemy

$$|A \cup B| = |A| + |B| - |A \cap B|, \quad (3.19)$$

a dzieląc obie strony przez $|\Omega|$ znajdujemy, zgodnie z 3.1

$$P(A \cup B) = P(A) + P(B) - P(A \cap B). \quad (3.20)$$



Rysunek 3.6: Diagramy Venna dla mających części wspólne dwóch, trzech i czterech zbiorów. Dla przypadku dwóch zbiorów zaznaczono też pełną przestrzeń Ω oraz elementy zbiorów (kropki)

Jest to bardzo ładny wzór, który mówi, że aby obliczyć prawdopodobieństwo zdarzenia $A \cup B$, dodajemy prawdopodobieństwa zdarzeń $P(A)$ i $P(B)$, ale od wyniku odejmujemy prawdopodobieństwo części wspólnej, $P(A \cap B)$. Z punktu widzenia definicji Laplace'a jest to oczywiste. Na przykładzie z lewej części rys. 3.6 zbiór A zawiera 4 elementy, a zbiór B 5 elementów, z czego 3 elementy są wspólne. Zatem dodając liczbę elementów A i B zliczylibyśmy wspólne elementy dwukrotnie. Aby skompensować to podwójne zliczanie odejmujemy elementy ze wspólnej części zbiorów $A \cap B$. Wtedy $5 + 4 - 3 = 6$, czyli dostajemy liczbę elementów w zbiorze $A \cup B$ i wszystko się zgadza.

Przykład 3.4. Rozważmy jako przykład obliczenie prawdopodobieństwa wyrzucenia parzystej lub podzielnej przez 3 liczby oczek w rzucie jedną kostką. W tym przypadku $A = \{\square, \blacksquare, \blacksquare\}$, $B = \{\square, \blacksquare\}$, $A \cap B = \{\blacksquare\}$ oraz

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = 3/6 + 2/6 - 1/6 = 4/6,$$

tyle, ile znajdujemy, licząc bezpośrednio moc zbioru $A \cup B = \{\square, \blacksquare, \blacksquare, \blacksquare\}$.

Dla przypadku trzech zbiorów ze środka rys. 3.6 sytuacja jest nieco bardziej skomplikowana. Licząc $|A| + |B| + |C|$, zliczamy podwójnie elementy należące do części wspólnych każdej pary zbiorów, które nie zawierają części wspólnej $A \cap B \cap C$ (obszary zaznaczone kolorem ciemnoszarym), natomiast elementy części wspólnej trzech zbiorów (kolor czarny) zliczamy potrójnie. Jeśli weźmiemy $|A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A|$, to skompensujemy podwójne zliczanie w obszarach ciemnoszarych, ale całkowicie wyeliminujemy zliczanie z obszaru czarnego. Jest tak, ponieważ $A \cap B$ jest sumą obszaru ciemnoszarego i czarnego. Zliczaliśmy potrójnie i odjęliśmy trzy razy, więc nic z obszaru czarnego nie pozostało! Musimy go teraz uwzględnić dodatkowo, wobec tego poprawny wzór to

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|.$$

Przykład 3.5. A teraz inny przykład, zahaczający o administrację państwową. Wyobraźmy sobie, że aby policzyć mieszkańców Polski, MSWiA⁶ zleciło wszystkim urzędom wojewódzkim policzenie mieszkańców, którzy w danym roku byli zameldowani w podległym urzędowi województwie. Zebrały liczby mieszkańców z wszystkich województw, dodano je do siebie, uzyskując wynik S_1 . Wynik ten nie jest jednak poprawny, ponieważ niektóre osoby zmieniły w trakcie roku zamieszkanie, np. przeprowadzając się z Krakowa do Kielc. Są wówczas ujęte zarówno w danych z Małopolski, jak i z woj. świętokrzyskiego. A zatem, aby skompensować tę nadwyżkę, MSWiA zastosowało następującą procedurę: zażądano list ewidencji mieszkańców z wszystkich województw, następnie brano wszystkie ich pary (mam nadzieję, że komputerowo) i zliczano, które osoby występują podwójnie, uzyskując w ten sposób liczbę S_2 . Następnie od S_1 odjęto S_2 , ale znowuż wynik nie jest dobry, bo są osoby, które mieszkały w trzech województwach, np. w Krakowie, Kielcach i Warszawie. W S_1 taka osoba jest policzona trzykrotnie, ale w S_2 też jest policzona trzykrotnie, przy porównywaniu list par województw małopolsko-świętokrzyskie, świętokrzyskie-mazowieckie i małopolskie-mazowieckie. W efekcie $S_1 - S_2$ w ogóle nie zawiera osób mieszkających w danym roku w trzech województwach! Trzeba je dodać, porównując trójki list. W ten sposób dostajemy S_3 i kolejny wynik dla liczby mieszkańców Polski: $S_1 - S_2 + S_3$. A co z tymi nadpotrzebnymi, którzy mieszkali w ciągu roku aż w czterech województwach? S_1 liczy ich 4 razy, S_2 odejmuje $\binom{4}{2} = 6$ razy, S_3 dodaje $\binom{4}{3} = 4$ razy, co daje w wyniku 2. Musimy więc odjąć liczbę S_4 , otrzymaną z porównania czwórek list ewidencji mieszkańców, dostając $S_1 - S_2 + S_3 - S_4$. Kontynuując to rozumowanie, dostaniemy wreszcie poprawny wynik $S_1 - S_2 + \dots - S_{16} + S_{17}$.

Ogólnie, dla dowolnej liczby zbiorów mamy następujące twierdzenie:

Tw. 3.4 (zasada włączania i wyłączania). Moc sumy zbiorów skończonych wyraża się wzorem

$$\begin{aligned} |\cup_{i=1}^n A_i| &= \sum_{i=1}^n |A_i| - \sum_{i>j=1}^n |A_i \cap A_j| + \sum_{i>j>k=1}^n |A_i \cap A_j \cap A_k| \\ &\quad - \sum_{i>j>k>l=1}^n |A_i \cap A_j \cap A_k \cap A_l| + \dots + (-1)^{n-1} |A_1 \cap A_2 \cap \dots \cap A_n|. \end{aligned} \quad (3.21)$$

Słownie, aby znaleźć moc sumy n zbiorów, od sumy mocy wszystkich tych zbiorów odejmujemy moce wszystkich możliwych przecięć (tj. części wspólnych) parzystej liczby zbiorów i dodajemy moce wszystkich możliwych przecięć nieparzystej liczby zbiorów. Ograniczenia „>” we wskaźnikach sumowania powodują, że zbiorów są

⁶ Ministerstwo Spraw Wewnętrznych i Administracji.

różne. Bardziej zwarty zapis tego samego stwierdzenia to

$$|\cup_{i=1}^n A_i| = \sum_I (-1)^{|I|-1} |\cap_{k \in I} A_k|, \quad (3.22)$$

gdzie użyliśmy symboli teoriomnogościowej sumy \cup i iloczynu \cap , natomiast suma po prawej stronie przebiega po wszystkich zbiorach indeksów I będących niepustymi podzbiorami zbioru wskaźników $\{1, 2, \dots, n\}$.

Dowód można przeprowadzić indukcyjnie. W szczególności dla czterech zbiorów mamy jawnie

$$\begin{aligned} |A \cup B \cup C \cup D| &= |A| + |B| + |C| + |D| \\ &- |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D| \\ &+ |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D| - |A \cap B \cap C \cap D|. \end{aligned}$$

Ogólną zasadę włączania i wyłączania dla prawdopodobieństw otrzymujemy poprzez podzielenie obu stron równania (3.22) przez $|\Omega|$:

$$P(\cup_{i=1}^n A_i) = \sum_I (-1)^{|I|-1} P(\cap_{k \in I} A_k). \quad (3.23)$$

3.4 Szaliki kibiców Korony Kielce

Przedstawimy teraz kolejny bardzo znany problem kombinatoryczny ilustrujący w nietrywialny sposób poznane wyżej metody, w szczególności zasadę włączania i wyłączania. Znany jest on jako „problem pomylonych kapeluszy” lub „pomyłonych parasoli”, który tutaj w trosce o lokalny koloryt przemianujemy na „problem szalików kibiców Korony Kielce”.

Def. 3.4. (problem szalików) Grupa n kibiców Korony Kielce, która zamiast na wykład matematyki dyskretnej poszła na mecz z Legią, rzuciła po strzelonej przez Koronę bramce swoje szaliki wysoko w górę! Jakie jest prawdopodobieństwo, że dokładnie k spośród rzucających złapie swój własny szalik? Zakładamy oczywiście, że szaliki spadają w sposób zupełnie losowy.

Szaliki mogą trafić do kibiców na $n!$ sposobów. Podzielmy te sposoby według kryterium, ile szalików, opadłszy na ziemię, trafiło do swoich właścicieli. Liczbę możliwości pomieszanego n szalików, gdzie dokładnie k z nich trafiło do właścicieli, oznaczmy przez K_k^n . Mamy C_k^n wyboru kombinacji szczęśliwych właścicieli. Pozostałych $n - k$ kibiców złapało cudze szaliki, a liczbę tych możliwości oznaczamy jako D_{n-k} . Liczbę D_m określa się też mianem liczby nieporządków, czyli takich permutacji m obiektów, które nie mają tzw. punktów stałych. W naszym przykładzie punktami stałymi są te szaliki, które wróciły do swoich prawowitych właścicieli.

Zgodnie z powyższą dyskusją

$$K_k^n = C_k^n D_{n-k}, \quad (3.24)$$

tj. liczba permutacji, gdzie k szalików wróciło do swoich właścicieli, to liczba wyborów k szczęścia spośród n kibiców razy liczba nieporządków dla pozostałych $n - k$ kibiców. Oczywiście, znalezienie rozwiązania dla K_k^n jest równoważne znalezieniu liczby nieporządków. Problem ten pierwszy raz postawił w 1708 r. i po pięciu latach rozwiązał (my, bogatsi o doświadczenia kilku epok, zrobimy to szybciej!) Pierre Raymond de Montmort⁷.

Przykład 3.6. Oto ilustracja problemu szalikowców na konkretnych liczbach. Dla $n = 5$ kibiców i $k = 2$ szczęśliwych kibiców mamy $C_2^5 = \binom{5}{2} = 10$ możliwości wyboru tych, co złapali swoje szaliki, natomiast D_3 jest liczbą takich permutacji pozostałych trzech szalików, że żaden z pechowych kibiców nie dostał swojego. Policzmy zatem te permutacje trójelementowe zbioru $\{1, 2, 3\}$, które nie pozostawiają żadnego elementu na swoim miejscu, czyli jedynka nie jest w pozycji pierwszej, dwójka w pozycji drugiej, a trójka w pozycji trzeciej. Są to właśnie trójelementowe nieporządkki. Wypisując wszystkie 6 możliwych permutacji, 123, 231, 312, 132, 321 i 213, zauważamy, że tylko dwie z nich, 231 oraz 312, nie pozostawiają żadnego elementu na swoim miejscu. Tak więc $D_3 = 2$ i $K_2^5 = C_2^5 D_3 = 10 \cdot 2 = 20$.

Do policzenia liczby nieporządków D_n zastosujemy regułę włączeń i wyłączeń. Wszystkich permutacji jest $n!$. Odejmujemy od nich wszystkie te, które mają któryś z liczb $1, \dots, n$ jako punkt stały. Wybrawszy konkretną liczbę na punkt stały, co możemy zrobić na n sposobów, pozostałe liczby możemy ustawić na $(n - 1)!$ sposobów, co daje łącznie $n(n - 1)!$. Wśród tych ustawień zdarzają się takie, które mają więcej niż jeden punkt stały, musimy więc dodać liczbę par punktów stałych razy liczba ustawień pozostałych $n - 2$ liczb, czyli $\binom{n}{2} (n - 2)!$. Kontynuując, wśród tych ustawień mamy takie, które mają więcej niż dwa punkty stałe, musimy więc odjąć $\binom{n}{3} (n - 3)!$ itd., co prowadzi do wzoru

$$\begin{aligned} D_n &= \sum_{k=0}^n \binom{n}{k} (-1)^k (n - k)! = \sum_{k=0}^n (-1)^k \frac{n!}{k!(n - k)!} (n - k)! = n! \sum_{k=0}^n \frac{(-1)^k}{k!} \\ &= n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!} \right). \end{aligned} \quad (3.25)$$

Prześledźmy powyższą procedurę zliczania dla przypadku $n = 3$. Wszystkich permutacji jest 6:

$$123, 132, 213, 231, 312, 321.$$

⁷ Pierre Raymond de Montmort (1678-1719), francuski matematyk.

Permutacje, gdzie 1, 2, lub 3 (oznaczone poniżej w nawiasie) są punktami stałymi (a inne elementy mogą być punktami stałymi lub nie), mają postać:

$$123, \quad 132, \quad (1)$$

$$123, \quad 321, \quad (2)$$

$$123, \quad 213. \quad (3)$$

Permutacje, gdzie 1 i 2, 1 i 3 oraz 2 i 3 (oznaczone w nawiasie) są punktami stałymi, to:

$$123, \quad (12)$$

$$123, \quad (13)$$

$$123. \quad (23)$$

Wreszcie, permutacja, gdzie wszystkie elementy są punktami stałymi, to

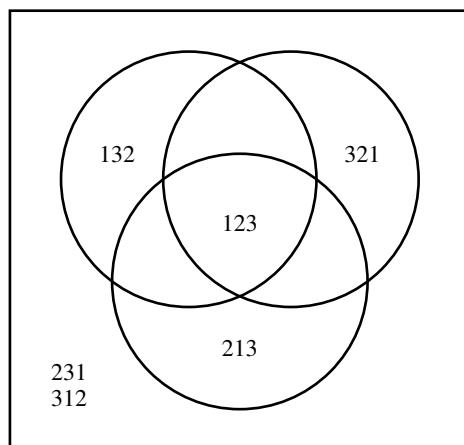
$$123. \quad (123)$$

Reguła włączeń i wyłączeń daje $D_3 = 6 - 3 \cdot 2 + 3 - 1 = 2$, co jest poprawnym wynikiem, gdyż mamy dwa nieporządkie

$$231, \quad 312.$$

Powyzsza analiza zobrazowana jest na rys. 3.7.

We wzorze (3.25) wyraziliśmy D_n w postaci skończonej sumy, co można uznać za rozwiązanie. Liczba sytuacji, gdzie k kibiców odzyskało swoje szaliki,



Rysunek 3.7: Diagram Venna dla podziału zbioru wszystkich permutacji zbioru $\{1, 2, 3\}$ na permutacje, których punktem stałym jest 1 (lewy górnny oval), 2 (prawy górnny oval) i 3 (dolny oval). Permutacja 123 jest wspólna dla tych trzech podzbiorów. Permutacje 231 i 312 nie mają punktów stałych (są nieporządkami)

dana jest wzorem (3.24). Ponieważ wszystkich możliwości jest $n!$, to prawdopodobieństwo, że k kibiców odzyskało swoje szaliki wynosi $P_k^n = K_k^n/n!$, czyli

$$P_k^n = \frac{1}{n!} \binom{n}{k} D_{n-k} = \frac{1}{k!(n-k)!} D_{n-k} = \frac{1}{k!} \sum_{l=0}^{n-k} \frac{(-1)^l}{l!}. \quad (3.26)$$

Jeśli $n = 12$, to dla kolejnych $k = 0, 1, \dots, 12$ dostajemy

$$0.37, 0.37, 0.2, 0.06, 0.02, 0.003, 0.0005, 0.00007, 9 \cdot 10^{-6}, 9 \cdot 10^{-7}, 10^{-7}, 0, 2 \cdot 10^{-9}.$$

Zauważmy, że dla $k = n - 1 = 11$ mamy dokładne zero. Ten fakt jest oczywisty, bo jest to sytuacja, gdzie dokładnie jeden kibic nie otrzymał swojego szalika. Ale jeśli on nie otrzymał swojego, to ktoś inny musiał dostać jego szalik, czyli też nie dostał swojego, a więc pokrzywdzonych, jeśli są, musi być co najmniej dwóch:

$$K_{n-1}^n = 0. \quad (3.27)$$

Zauważmy, że najczęściej jest sytuacji, gdzie nikt nie odzyskuje szalika, ok. 37%, niemal tyle samo (różnica względna jest dla $n = 12$ rzędu 10^{-9}), kiedy jeden kibic dostaje swój szalik. Dla większych k prawdopodobieństwa są wyraźnie coraz mniejsze. Fakt, że dla większych n prawdopodobieństwo P_0^n jest bardzo bliskie P_1^n , wynika stąd, że $P_0^n - P_1^n = (-1)^n/n!$, co wynika natychmiast z (3.26). Innymi słowy

$$K_0^n - K_1^n = (-1)^n. \quad (3.28)$$

Zajmijmy się teraz bardziej szczegółowo przypadkiem $k = 0$, tj. gdzie nikt nie odzyskał szalika. Mamy

$$P_0^n = \frac{D_n}{n!} = \sum_{l=0}^n \frac{(-1)^l}{l!}. \quad (3.29)$$

Ponieważ

$$e^x = \sum_{l=0}^{\infty} \frac{x^l}{l!},$$

biorąc $x = -1$ dostajemy natychmiast z (3.29)

$$\lim_{n \rightarrow \infty} P_0^n = \frac{1}{e} \simeq 0.368.$$

Na koniec popatrzmy jeszcze na liczbowe wyniki dla liczb szalikowych K_k^n , zebrane w tabeli 3.1. Widzimy w niej wszystkie dyskutowane wcześniej cechy: $K_0^n = 1$ – jest tylko jedna możliwość złapania szalików, gdzie wszystkie trafiają do właścicieli, ponadto mamy $K_{n-1}^n = 0$, bo niemożliwa jest sytuacja, w której tylko jeden kibic dostaje cudzy szalik, ponadto widzimy związek $K_0^n - K_1^n = (-1)^n$ – dla parzystych n najczęściej jest przypadków, gdzie nikt nie odzyskuje swojego szalika, a dla n nieparzystych, gdy szalik odzyskuje dokładnie jeden właściciel.

Inna metoda rozwiązania problemu przedstawiona jest szczegółowo w [3].

Tabela 3.1: Liczby szalikowe K_k^n , czyli liczba n -elementowych permutacji o k punktach stałych

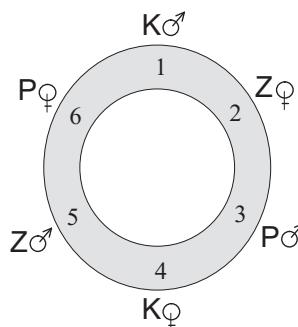
n	$k = 0$	1	2	3	4	5	6	7
0	1							
1	0	1						
2	1	0	1					
3	2	3	0	1				
4	9	8	6	0	1			
5	44	45	20	10	0	1		
6	265	264	135	40	15	0	1	
7	1854	1855	924	315	70	21	0	1
8	14833	14832	7420	2464	630	112	28	0
9	133496	133497	66744	22260	5544	1134	168	36

3.5 Problem sadowienia gości

Jako kolejną ilustrację nietrywialnego wykorzystania zasady włączania i wyłączania 3.21 podamy rozwiązanie pewnego dość zaawansowanego problemu kombinatorycznego związanego z organizowaniem eleganckich przyjęć.

Problem 3.2 (problem sadowienia). *Na ile M_n sposobów można posadzić przy okrągłym stole (z rozróżnialnymi miejscami) n par małżeńskich tak, aby mężczyźni siedzieli na przemian z kobietami, a nikt nie siedział koło swojego współmałżonka?*

Metoda rozwiązania przedstawiona poniżej pochodzi z pracy [41]. Wszystkich możliwości usadnień jest $|\Omega| = 2(n!)^2$: czynnik 2, bo kobiety mogą siedzieć na krzesłach o numerach parzystych lub nieparzystych, ponadto mamy $n!$ możliwości posadzenia kobiet i drugie $n!$ możliwości posadzenia mężczyzn. Rozważmy na początek najmniejszą liczbę par, dla której zadanie jest wykonalne, $n = 3$. Mamy więc trzy pary: Kowalscy, Zielińscy i Przybylscy oraz $2(3!)^2 = 72$ możliwości wszystkich usadnień. Przykładowa sytuacja pokazana jest na rys. 3.8.



Rysunek 3.8: Przykładowe usadzenie trzech par, gdzie mężczyźni siedzą na przemian z kobietami i nikt nie siedzi koło swojego współmałżonka. Kowalski siedzi na krześle 1, Zielińska na krześle 2 itd.

A teraz zasada włączania i wyłączania w akcji: podzielmy zbiór wszystkich usadowień Ω na podzbiory tak, jak na rys. 3.9, który przedstawia diagram Venna dla podziału zbioru Ω na podzbiory względem kryterium, które pary siedzą koło siebie. Niech K oznacza zbiór wszystkich usadowień, w których Kowalscy siedzą koło siebie, Z – „Zielińscy koło siebie”, P – „Przybylscy koło siebie”. Obszar na zewnątrz kół odpowiada podzbiorowi W , gdzie nikt nie siedzi koło swojego współmałżonka. Mamy więc zgodnie z zasadą włączeń i wyłączeń

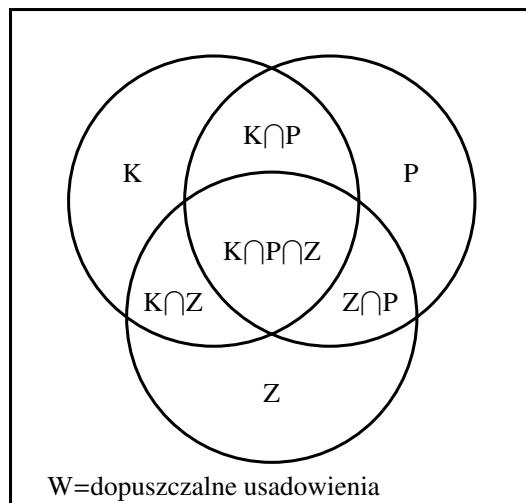
$$|W| = |\Omega| - |K| - |P| - |Z| + |K \cap P| + |K \cap Z| + |Z \cap P| - |K \cap P \cap Z|.$$

Ponieważ $|K| = |P| = |Z|$ (liczba możliwości usadowień nie zależy od tego, która para siedzi razem) oraz analogicznie $|K \cap P| = |K \cap Z| = |Z \cap P|$, możemy też zapisać

$$\begin{aligned} |W| &= |\Omega| - 3|K| + 3|K \cap P| - |K \cap P \cap Z| \\ &= \binom{3}{0}|\Omega| - \binom{3}{1}|K| + \binom{3}{2}|K \cap P| - \binom{3}{3}|K \cap P \cap Z|, \end{aligned} \quad (3.30)$$

gdzie w ostatniej linii zapisaliśmy współczynniki w postaci uwzględniającej kombinatorykę, np. jeden element z trzech możemy wybrać na $\binom{3}{1}$ sposobów itp.

Pora na uogólnienie dla dowolnego n . W tym celu wprowadźmy oznaczenie A_k dla zbioru usadowień, w których k wybranych par siedzi razem, a inne mogą siedzieć razem lub nie. Dla przykładu trzech par, $A_0 = \Omega$, $A_1 = K$, $A_2 = |K \cap P|$



Rysunek 3.9: Diagram Venna dla podziału zbioru wszystkich usadowień trzech par na podzbiory. K oznacza „Kowalscy siedzą koło siebie”, $K \cap Z$ – „Kowalscy oraz Zielińscy siedzą koło siebie” itd. Obszar na zewnątrz kół odpowiada podzbiorowi W właściwych usadowień na przyjęciu, gdzie nikt nie siedzi koło swojego współmałżonka

i $A_3 = |K \cap P \cap Z|$. Uogólnienie wzoru (3.30) ma postać

$$M_n = |W| = \sum_{k=0}^n (-1)^k \binom{n}{k} |A_k|, \quad (3.31)$$

gdzie z oznaczenia $M_n = |W|$. Pozostaje wyznaczyć $|A_k|$, czyli liczbę możliwości usadówień, w których k wybranych par siedzi razem (a inne mogą siedzieć razem lub nie). Podstawowa obserwacja jest teraz następująca: liczba możliwości wyboru miejsc dla k wybranych par to liczba pokryć $2n$ punktów na okręgu za pomocą k kości domina i $2n - k$ kwadratów – patrz wcześniej rozwiązany problem 2.13 i rys. 2.13 – co daje c_k^{2n} możliwości ze wzoru (2.35). Zliczanie dla $|A_k|$ jest więc następujące:

- Kobiety na parzystych lub nieparzystych krzesłach – 2.
- Wybór (podwójnych) miejsc dla wybranych k par – $c_k^{2n} = \frac{2n}{2n-k} \binom{2n-k}{k}$, to samo, co liczba pokryć $2n$ punktów na okręgu za pomocą k kości domina i $2n - k$ kwadratów.
- Rozmieszczenie k par w k miejscach – $k!$.
- Usadzenie pozostałych $n - k$ kobiet – $(n - k)!$.
- Usadzenie pozostałych $n - k$ mężczyzn – $(n - k)!$.

Mamy więc

$$|A_k| = 2c_k^{2n} k! (n - k)!^2 = 2 \frac{2n}{2n - k} \frac{(2n - k)!}{(2n - 2k)!} (n - k)!^2$$

Tabela 3.2: Liczba usadówień n mażeństw na przyjęciu

n	M_n	$M_n / (2n!)^2$
1	0	0
2	0	0
3	12	1/6
4	96	1/12
5	3120	0.108
6	115200	0.111
7	5836320	0.115
8	382072320	0.118
9	31488549120	0.120
10	3191834419200	0.121
100	$2.3 \cdot 10^{315}$	0.134
1000	$4.4 \cdot 10^{5134}$	0.135

oraz ostatecznie szukany wynik w postaci sumy:

$$\begin{aligned} M_n &= \sum_{k=0}^n (-1)^k \binom{n}{k} 2 \frac{2n}{2n-k} \frac{(2n-k)!}{(2n-2k)!} (n-k)!^2 \\ &= 2n! \sum_{k=0}^n (-1)^k 2 \frac{2n}{2n-k} \binom{2n-k}{k} (n-k)!. \end{aligned} \quad (3.32)$$

Kilka pierwszych wartości dla liczb M_n ze wzoru (3.32) oraz dla prawdopodobieństwa uzyskania losowo właściwego usłownienia, $M_n/|\Omega| = M_n/(2n!^2)$, przedstawionych jest w tabeli 3.2.

3.6 Prawdopodobieństwo warunkowe

Często w zagadnieniach rachunku prawdopodobieństwa mamy do czynienia z sytuacją, w której posiadamy dodatkowe informacje zawężające przestrzeń zdarzeń. Na przykład ktoś każe nam zgadywać, ile oczek wypadło mu w rzucie kostką. Jeśli nie mamy żadnych dodatkowych informacji, to prawdopodobieństwo, że zgadniemy poprawnie wynosi $1/6$. Jeśli jednak rzucający powiedział nam, że wypadła parzysta liczba oczek, to wtedy mamy ułatwione zadanie i prawdopodobieństwo poprawnej odpowiedzi to $1/3$, bo mogły jedynie wypaść \square , \blacksquare lub $\blacksquare\blacksquare$. Prawdopodobieństwo zdarzenia A w sytuacji, gdzie przestrzeń zdarzeń jest zawężona do zbioru B oznaczamy jako $P(A|B)$ i czytamy „prawdopodobieństwo zajścia A pod warunkiem zajścia B ”. Odpowiednik definicji Laplace'a dla prawdopodobieństwa warunkowego jest więc następujący:

Def. 3.5. *Prawdopodobieństwo warunkowe zajścia zdarzenia A pod warunkiem zajścia zdarzenia B definiujemy jako*

$$P(A|B) = \frac{|A \cap B|}{|B|}.$$

Prostego przykładu dostarcza lewa część rys. 3.6. Jakie jest prawdopodobieństwo wylosowania elementu ze zbioru A pod warunkiem, że element należy do zbioru B ? Zbiór B zawiera łącznie 5 elementów, z czego 3 należą też do A , w związku z tym $P(A|B) = |A \cap B|/|B| = 3/5$. Podobnie, prawdopodobieństwo zajścia zdarzenia B pod warunkiem zajścia zdarzenia A wynosi $P(B|A) = |A \cap B|/|A| = 3/4$.

Podamy teraz użyteczne twierdzenie dotyczące sytuacji, gdy przestrzeń zdarzeń Ω rozpada się na zbiory rozłączne.

Tw. 3.5. *(o prawdopodobieństwie całkowitym) Jeżeli $\Omega = A_1 \cup A_2 \cup \dots \cup A_n$, i zbiory A_i są parami rozłączne, to*

$$P(B) = P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + \dots + P(B|A_n)P(A_n).$$

Dowód: Z definicji mamy

$$\begin{aligned} P(B) &= \frac{|B \cap A_1|}{|A_1|} \frac{|A_1|}{|\Omega|} + \cdots + \frac{|B \cap A_n|}{|A_n|} \frac{|A_n|}{|\Omega|} \\ &= \frac{1}{|\Omega|} |(B \cap A_1) \cup \cdots \cup (B \cap A_n)| = \frac{1}{|\Omega|} |B \cap (A_1 \cup \cdots \cup A_n)| \\ &= \frac{1}{|\Omega|} |B \cap \Omega| = \frac{|B|}{|\Omega|} = P(B), \end{aligned}$$

gdzie użyliśmy faktu, że suma mocy zbiorów rozłącznych $B \cap A_i$ jest równa mocy sumy tych zbiorów, oraz skorzystaliśmy z definicji Laplace'a. \square

3.7 Twierdzenie Bayesa

Z prawdopodobieństwami warunkowymi związane jest słynne Tw. Bayesa⁸

Tw. 3.6. *Prawdopodobieństwa warunkowe spełniają zależność*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Dowód: Definicję 3.5 możemy przepisać w następujący sposób:

$$P(A|B) = \frac{|A \cap B|}{|B|} = \frac{|A \cap B|/|\Omega|}{|B|/|\Omega|} = \frac{P(A \cap B)}{P(B)},$$

skąd $P(A \cap B) = P(A|B)P(B)$. Analogicznie, zamieniając rolami A i B , otrzymujemy $P(B \cap A) = P(B|A)P(A)$. Przyciągając stronami daje $P(A|B)P(B) = P(B|A)P(A)$, co prowadzi do tezy. \square

Wygodnie jest niekiedy używać nieco innego zapisu Tw. Bayesa. Ponieważ

$$P(B) = P(B \cap (A \cup \bar{A})) = P(B \cap A) + P(B \cap \bar{A}) = P(B|A)P(A) + P(B|\bar{A})P(\bar{A}),$$

mamy

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})}. \quad (3.33)$$

Chociaż dowód Tw. Bayesa jest oczywisty czy wręcz banalny, samo twierdzenie ma daleko idące konsekwencje. Wprowadźmy teraz pewną nomenklaturę dopasowaną do typowych zastosowań. Prawdopodobieństwo zajścia zdarzenia A bez żadnych dodatkowych warunków nazywamy prawdopodobieństwem *a priori* (tzn. z góry, z założenia), a prawdopodobieństwo warunkowe $P(A|B)$ prawdopodobieństwem *a posteriori* (przeciwieństwo *a priori*), czyli po uzyskaniu informacji o zajściu zdarzenia B .

⁸ Thomas Bayes (1702-1761), angielski pastor, autor pracy *An Essay Towards Solving a Problem in the Doctrine of Chances*, w której m.in. sformułował swoje twierdzenie.

Inna użyteczna postać twierdzenia Bayesa pozwala porównać dwa prawdopodobieństwa *a posteriori*. Pisząc bowiem Tw. 3.6 dla zdarzeń A i C oraz B i C , a następnie dzieląc stronami, otrzymujemy proporcję

$$\frac{P(A|C)}{P(B|C)} = \frac{P(C|A)}{P(C|B)} \frac{P(A)}{P(B)}. \quad (3.34)$$

Przykład 3.7. Założmy, że na stole stoją dwie bardzo podobne patery z owocami. Na jednej jest 6 jabłek i 12 gruszek, a na drugiej 9 jabłek i 9 gruszek. Babcia poała Józio po jakiś owoc i wnuk przyniósł jabłko. Jakie jest prawdopodobieństwo, że Józio wziął owoc z pierwszej patery?

A priori Józio wybiera każdą z paterek z prawdopodobieństwem $1/2$, bo patery są podobne i Józio nie robi to żadnej różnicę. Bez dodatkowej informacji prawdopodobieństwo wybrania pierwszej patery wynosi $P(I) = 1/2$, tudzież prawdopodobieństwo wybrania drugiej patery wynosi $P(II) = 1/2$. Prawdopodobieństwo a priori wybrania jabłka możemy zapisać jako $P(j) = P(j|I)P(I) + P(j|II)P(II) = 6/18 \cdot 1/2 + 9/18 \cdot 1/2 = 5/12$. Babcia wie jednak, ile owoców było na każdej z paterek, a teraz wnuk przyniósł jabłko. W związku z tym, po uzyskaniu tej jakże cennej dodatkowej informacji babcia stosuje wzór Bayesa i dostaje $P(I|j) = P(j|I)P(I)/P(j) = 6/18 \cdot 1/2/(5/12) = 2/5$ oraz $P(II|j) = P(j|II)P(II)/P(j) = 9/18 \cdot 1/2/(5/12) = 3/5$. Wynik zgadza się z intuicją. Na pierwszej paterze było względnie mniej jabłek, więc szansa, że jabłko pochodzi z tej patery jest mniejsza od $1/2$.

A teraz przykład, który jest znacznie mniej intuicyjny.

Przykład 3.8. Hipochondryk chce się zdiagnozować, czy nie ma przypadkiem choroby wściekłych krów. Założmy, że test na obecność tej choroby daje u chorych Ch wynik pozytywny + z prawdopodobieństwem $P(+|Ch) = 0.99$, a u zdrowych Z wynik negatywny – z prawdopodobieństwem $P(-|Z) = 0.99$. Powiedzielibyśmy, że testy są bardzo dobre, bo mylą się tylko w 1% przypadków. Założymy też, że prawdopodobieństwo zapadnięcia na chorobę wynosi $P(Ch) = 10^{-6}$, czyli raz na milion – choroba wściekłych krów jest w istocie rzadka. Policzymy zgodnie ze wzorem Bayesa prawdopodobieństwo, że osoba jest chora, jeśli test daje wynik pozytywny. Mamy

$$\begin{aligned} P(Ch|+) &= \frac{P(+|Ch)P(Ch)}{P(+|Ch)P(Ch) + P(+|Z)P(Z)} \\ &= \frac{P(+|Ch)P(Ch)}{P(+|Ch)P(Ch) + (1 - P(-|Z))(1 - P(Ch))} \\ &= \frac{0.99 \cdot 10^{-6}}{0.99 \cdot 10^{-6} + 0.01 \cdot (1 - 10^{-6})} \simeq 10^{-4}, \end{aligned} \quad (3.35)$$

tym samym prawdopodobieństwo, że się jest zdrowym, gdy test wyszedł pozytywnie, wynosi

$$P(Z|+) = 1 - P(Ch|+) \simeq 1.$$

Hipochondryk zapewne niezle się przerazi, jeśli test wyjdzie dodatnio, natomiast jak widzimy z powyższej analizy, nie ma się zupełnie czym przejmować! Gdzie tkwi swoisty paradoks? Popatrzmy dokładniej. Zażądamy, by

$$P(Ch|+) > u, \quad (3.36)$$

czyli pragniemy mieć więcej niż u pewności choroby przy dodatnim teście. Jednocześnie wynika stąd, że

$$P(Z|+) = 1 - P(Ch|+) < 1 - u. \quad (3.37)$$

Zastosujmy teraz wzór (3.34):

$$\frac{P(+|Z)}{P(+|Ch)} \frac{P(Z)}{P(Ch)} = \frac{P(Z|+)}{P(Ch|+)} < \frac{1-u}{u}, \quad (3.38)$$

gdzie nierówność wynika ze wzorów (3.36, 3.37). Dla rzadkiej choroby $P(Z) = 1 - P(Ch) \simeq 1$, ponadto test daje dla chorych niemal zawsze wynik pozytywny, $P(+|Ch) \simeq 1$. Wówczas nierówność (3.38) można zapisać jako

$$P(+|Z) < \frac{1-u}{u} P(Ch). \quad (3.39)$$

Co oznacza ten warunek? Na przykład dla $u = 90\%$ mówi, że prawdopodobieństwo przekłamania, czyli stwierdzenia dodatniego testu u osoby zdrowej, powinno być mniejsze niż $1/9$ częstotliwości występowania choroby. Dla bardzo rzadkich chorób oznacza to w praktyce, że test nie może przekłamywać, czyli nie może dawać pozytywnego wyniku dla osoby zdrowej. Tylko takie testy są w tym przypadku diagnostycznie użyteczne. Dla przyjętej wyżej częstotliwości występowania choroby równej 10^{-6} , test może przekłamywać na poziomie co najwyżej 10^{-7} . Natomiast test na chorobę wściekłych krów, któremu poddał się hipochondryk, przekłamywał aż w 1% – w tym przypadku 1% to bardzo dużo, 10^4 razy za dużo, gdy porównamy z częstotliwością występowania choroby.

Inne spojrzenie na powyższy przykład jest następujące. Weźmy populację Polski, w zaokrągleniu 40 mln, i wyobraźmy sobie, że wszyscy poddani są naszemu testowi. Statystycznie więc będzie $10^{-6} \cdot 40 \text{ mln} = 40$ chorych, a test dokładny na poziomie 99% da dla nich wynik pozytywny. Natomiast przekłamywanie testu dla osób zdrowych przyniesie wynik pozytywny dla ok. $1\% \cdot 40 \text{ mln} = 400$ tys. osób. Wobec tego wśród wszystkich przebadanych z wynikiem pozytywnym będzie 40 chorych i 400 tys. zdrowych! To jest sens wzoru (3.35) interpretowanego „na chłopski rozum”.

Natomiast dla bardziej popularnych „przypadłości” dopuszczalny jest pewien poziom przekłamywań. Rozważmy test ciążowy. Niech prawdopodobieństwo a

priori ciąży u kobiety poddającej się testowi wynosi 20%. Wtedy test stwierdzający odmienny stan kobiety nie w ciąży (przekłamanie) z prawdopodobieństwem na poziomie 1% jest nadal użyteczny, jak wynika ze wzoru (3.39).

Ćwiczenia

- 3.1. A oto wspomniany problem Antoine Gombauda, kawalera de Méré. Rzucamy dwiema kości 24 razy pod rząd. Jakie jest prawdopodobieństwo, że choć raz wypadną dwie jedynki, $\square \square$?
- 3.2. Rozważ tzw. kości Efrona⁹. Są to sześciennne kości do gry, w których poszczególne ściany posiadają następujące *niestandardowe* liczby oczek:

I:	4,4,4,4,0,0
II:	3,3,3,3,3,3
III:	6,6,2,2,2,2
IV:	5,5,5,1,1,1

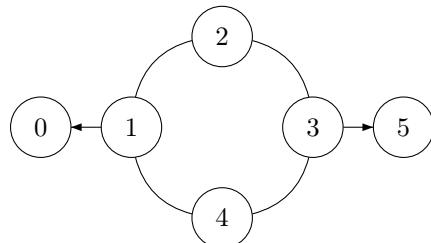
Czterej gracze, każdy z kością I, II, III lub IV, grają ze sobą parami i wygrywa ten, który wyrzuci większą liczbę oczek. Jakie jest prawdopodobieństwo, że I wygra z II, II z III itd.? Przedyskutuj wynik.

- 3.3. Która z kości z ćw. 1 jest „najlepsza”, tzn. w grze z losowo wybranym przeciwnikiem daje największe prawdopodobieństwo wygranej?
- 3.4. Jakie są prawdopodobieństwa wylosowania (w jednym rozdaniu) następujących układów pokera: para, dwie pary, strit, trójka, full, kolor, kareta, poker? Załóż, że talia jest od a) dziewczątka, b) siódemek, c) dwójków. Wybierz do rozwiązania kilka przypadków.
- 3.5. Trzej równej klasy szachiści A , B i C grają w szachy według zasady, że gracz wygrywający dwie partie z rzędu wygrywa turniej, a gracz przegrywający partię w następnej pauzuje. Zaczynają A z B . Gracze grają do skutku. Jakie jest prawdopodobieństwo wygrania turnieju przez poszczególnych graczy?
- 3.6. Pokaż, że dla problemu szalików $\bar{k} = \sum_{k=0}^n k P_k^n = 1$. Wynik ten oznacza, że średnia liczba kibiców, którzy odzyskali swój szalik, wynosi 1.
- 3.7. Rozwiąż zmodyfikowany problem sadowienia na przyjęciu, gdzie żądamy, aby współmałżonkowie nie siedzieli koło siebie, ale kobiety i mężczyźni *nie muszą* siedzieć na przemian.
- 3.8. Założmy, że po Kielcach jeździ 12% taksówek zielonych i 88% niebieskich. Roztargniony profesor zostawił laptopa w podwożącej go wieczorem na uniwersytet taksówce. Portier, który po ciemku rozróżnia kolor niebieski od zielonego z prawdopodobieństwem 80% (tzn. z prawdopodobieństwem 80% określa taksówkę zieloną jako zieloną i z prawdopodobieństwem 20% określa taksówkę niebieską jako zieloną), zeznał, że widział zieloną taksówkę. Jakie

⁹ Bradley Efron (1938 –), amerykański statystyk.

jest prawdopodobieństwo, że profesor zostawił laptopa w zielonej taksówce?

- 3.9. Podczas targów zbrojeniowych w Kielcach firma „Łucznik” urząduje następujący konkurs promocyjny dla generałów: na scenie znajdują się trzy zamknięte pudła, przy czym jedno z nich kryje karabin „Beryl”, a dwa są puste. Prowadzący wie, w którym pudle jest „Beryl”. Generał oczywiście tego nie wie i zgaduje, jednak zanim wskazane pudło zostaje otwarte, prowadzący otwiera jedno z pozostałych dwóch pudeł, które jest puste (wie, które są puste) i mówi: „Panie generale, może Pan jeszcze zmienić swój wybór!”. Czy generał powinien zmienić swoje pierwotne wskazanie?
- 3.10. Mysz chodzi zupełnie losowo po labiryncie z rys. 3.10, przechodząc z każdego pomieszczenia o numerze 1, 2, 3 lub 4 do sąsiadniego (połączonego korytarzem) z równym prawdopodobieństwem. Jeśli dojdzie do pomieszczenia 0, to ginie, a jeśli dojdzie do pomieszczenia 5, to wychodzi na wolność. (a) Jakie jest prawdopodobieństwo wyjścia myszy na wolność, jeśli początkowo znajduje się w pomieszczeniu 2? (b) Wiedząc, że laborant włożył mysz losowo do któregoś z pomieszczeń 1, 2, 3 lub 4, oraz że mysz uciekła, jakie jest prawdopodobieństwo, że została włożona do pomieszczenia i ?



Rysunek 3.10: Diagram błędzenia myszy z ćw. 3.10

Rozdział 4

Więcej kombinatoryki

W tym rozdziale powracamy do bardziej zaawansowanej kombinatoryki, prowadzącej do znanych rodzin liczb mających szerokie zastosowanie w problemach zliczania. Zaczniemy od grupy permutacji, by kolejno omówić liczby Stirlinga, liczby Bella oraz partycje liczb na składniki naturalne. Zakończymy nasze rozważania, podając kompendium najczęściej pojawiających się w praktyce problemów kombinatorycznych.

4.1 Grupa permutacji

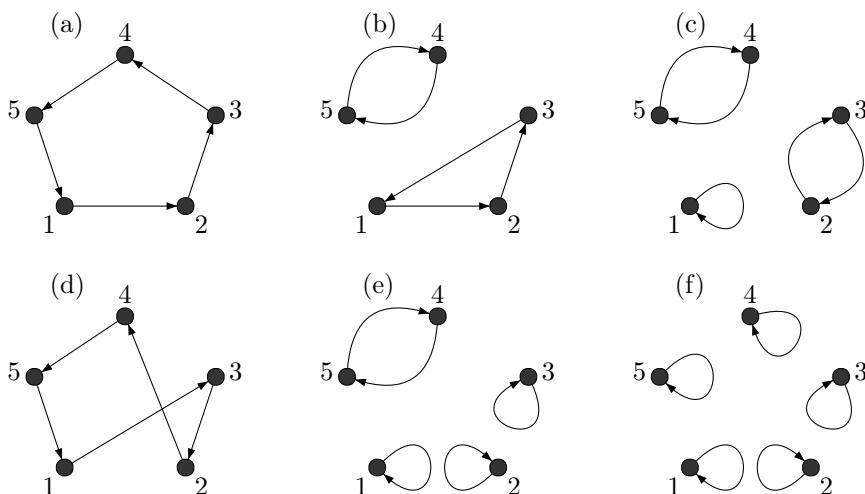
Permutację n -elementowego zbioru zdefiniowaliśmy już w (2.1), natomiast szczegółowo użytkownicza jest równoważna definicja, traktująca permutację jako odwzorowanie zbioru na siebie (bijekcję). Co robi permutacja? Po prostu zmienia kolejność ustawienia przedmiotów. W oczywisty sposób odwzorowanie to możemy składać oraz odwracać: możemy kilka razy zmienić kolejność ustawienia, możemy też dla każdej zmiany wykonać zmianę odwrotną. Istnieje też odwzorowanie identycznościowe, które nie dokonuje żadnej zmiany kolejności. Widzimy więc, że permutacje tworzą grupę, którą oznaczamy S_n i nazywamy grupą symetryczną n obiektów. Zbiorem jest tu zbiór wszystkich bijekcji zbioru n -elementowego na siebie, a działaniem operacja składania funkcji. Ponieważ zbiór n -elementowy

możemy utożsamić ze zbiorem n kolejnych liczb naturalnych, $\{1, 2, 3, \dots, n\}$, będącymi dalej rozważać permutacje tego zbioru.

Do oznaczania konkretnej permutacji stosuje się dwa podstawowe zapisy. Pierwszy z nich podaje funkcję zadaną tabelką, np. zapis

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 3 & 1 \end{pmatrix} \quad (4.1)$$

oznacza przyporządkowanie $1 \rightarrow 2, 2 \rightarrow 4, 3 \rightarrow 5$ itd. Drugi zapis związany jest ściśle z reprezentacją graficzną. Rysunek 4.1 przedstawia kilka wybranych permutacji zbioru pięcioelementowego. Strzałki pokazują jak działa odwzorowanie, np. permutacja z rys. 4.1(a) przeprowadza $1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5$, wreszcie $5 \rightarrow 1$, co możemy zapisać (12345) – kolejnemu elementowi przyporządkowany jest następny na liście, a ostatniemu pierwszy. Moglibyśmy też zapisać tę samą permutację jako np. (23451) , ale wygodnie jest przyjąć konwencję, że zaczynamy od najmniejszej liczby. Drugi przedstawiony zapis jest bardziej zwarty od pierwszego i więcej nam mówi o strukturze danej permutacji. W szczególności zauważamy natychmiast, że permutacje mogą się rozпадać na rozłączne pętle, tzw. *cykle*, co oznaczamy nawiasami. Na przykład permutacja z rys. (b) to $(123)(45)$, czyli składa się z cykli o długościach 3 i 2. Rys. (a) i (d) obrazują permutacje o jednym cyklu. Rysunek na dole po prawej przedstawia permutację identycznościową $e = (1)(2)(3)(4)(5)$, składającą się z pięciu cykli o długości 1, tzw. cykli *singletowych*. Możemy jeszcze skrócić notację poprzez opuszczenie cykli singletowych. Istotnie, jeśli zapis nie zawiera jakiegoś elementu, to wnioskujemy, że stanowi on cykl singletowy. W szczególności permutacje z rys. 4.1 zapisujemy jako $(12345), (123)(45), (23)(45), (13245), (1)(2)(3)(45)$ oraz e .



Rysunek 4.1: Przykładowe elementy grupy S_5 permutacji zbioru pięcioelementowego: $(12345), (123)(45), (1)(2)(3)(45), (13245), (1)(2)(3)(45)$ oraz identycznościowa $(1)(2)(3)(4)(5) = e$

Za pomocą wprowadzonego zapisu łatwo jest składać (mnożyć) permutacje. Na przykład złożenie permutacji z rys. 4.1(b) (oznaczmy ją jako p_b) i z rys. 4.1(a) (p_a) zapisujemy jako $p_a \circ p_b = (12345) \circ (123)(45)$. Następnie wykonujemy następującą procedurę: zaczynamy od cyklu najbardziej po prawej stronie permutacji p_b , gdzie $4 \rightarrow 5$. Z kolei permutacja p_a przyporządkowuje $5 \rightarrow 1$, więc piszemy początek wyniku mnożenia: $p_a \circ p_b = (41\dots$, po czym idziemy dalej, biorąc ostatni wypisany element, czyli 1. Permutacja p_b odwzorowuje $1 \rightarrow 2$, a permutacja p_b $2 \rightarrow 3$, mamy więc już $p_a \circ p_b = (413\dots$. Następnie element 3 jest odwzorowywany przez p_b w 1, a potem 1 przez p_a w 2, co daje częściowy wynik $p_a \circ p_b = (4132\dots$. Dalej, 2 jest odwzorowywane przez p_b w 3, a 3 przez p_a w 4. Ponieważ element 4 został już wypisany na początku tworzonego wyniku, zgodnie z naszą konwencją dopisujemy teraz tylko nawias, $p_a \circ p_b = (4132)\dots$, co oznacza właśnie, że liczba 2 przyporządkowana jest liczba 4. Po zamknięciu cyklu patrzymy, który element jeszcze nie wystąpił w dotychczas uzyskanym wyniku. W naszym przypadku jest to 5. Powtarzamy teraz powyższy algorytm dla tego elementu. Permutacja p_b przeprowadza $5 \rightarrow 4$, a p_a $4 \rightarrow 5$, tak więc otrzymujemy cykl singletowy i ostatecznie $p_a \circ p_b = (4132)(5)$. Możemy jeszcze opuścić znak \circ w mnożeniu oraz cykl singletowy, co daje ostatecznie $p_a p_b = (4132)$.

Zauważmy jeszcze, że dla $n > 2$ grupa permutacji S_n nie jest przemienna. Dla $n = 3$ kontrprzykładem jest $(12)(23) = (123)$, podczas gdy $(23)(12) = (132)$. Każda grupa S_n o $n > 3$ zawiera S_3 jako podgrupę, zatem też nie jest przemienna.

Więcej informacji o grupie symetrycznej i permutacjach można znaleźć np. w [42, 43].

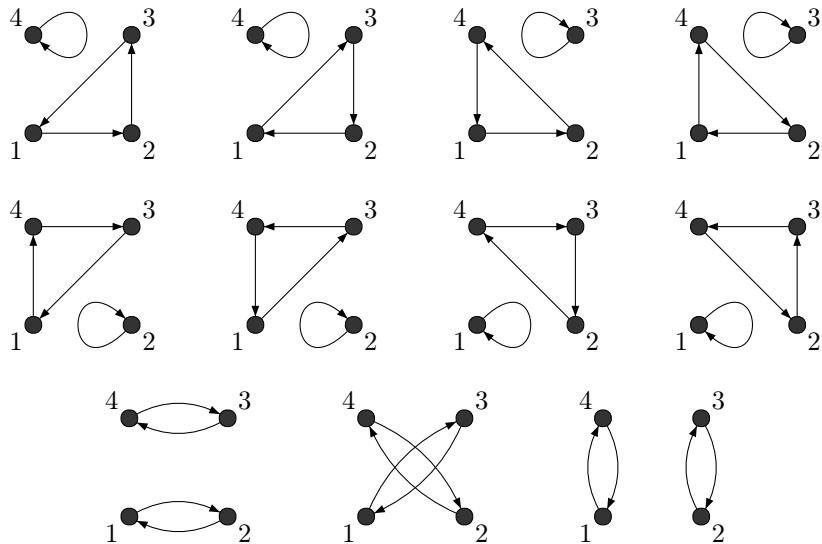
4.2 Liczby Stirlinga

Posiadłszy elementarne wiadomości o grupie permutacji i cyklach, możemy rozważyć następujący problem kombinatoryczny:

Problem 4.1 (tańce w kółku). *Podczas praktyki w przedszkolu student informatyki przygląda się n tańczącym dzieciom, które połączyły się w kółek, trzymając się w taki sposób, że każda prawa ręka uchwyciła jedną lewą. Student zastanawia się, na ile sposobów można to zrobić (pojedyncze dziecko tworzy „kółko” samo ze sobą).*

Mówiąc matematycznie, wśród permutacji n obiektów, ile z nich zawiera dokładnie k cykli? Na przykład dla $n = 4$ i $k = 2$ mamy następujące cykle: $(123)(4)$, $(132)(4)$, $(124)(3)$, $(142)(3)$, $(143)(2)$, $(134)(2)$, $(243)(1)$, $(234)(1)$, $(12)(34)$, $(13)(24)$, $(14)(23)$, czyli łącznie 11 możliwości. Sytuacja ukazana jest na rys. 4.2. W stylistyce problemu 4.1, strzałka wychodząca z punktu i oznacza kierunek wyciągniętej prawej ręki dziecka numer i .

Rozwiążanie tego problemu określa kolejne ciekawe liczby o kombinatorycznej interpretacji: *liczby Stirlinga pierwszego rodzaju*.



Rysunek 4.2: Wszystkie 11 permutacji zbioru czteroelementowego o dokładnie dwóch cyklach. Liczba możliwości to liczba Stirlinga pierwszego rodzaju

Def. 4.1. *Liczba Stirlinga pierwszego rodzaju, oznaczona jako*

$$\left[\begin{array}{c} n \\ k \end{array} \right],$$

to liczba n -elementowych permutacji o dokładnie k cyklach¹.

Tw. 4.1. *Liczby Stirlinga pierwszego rodzaju spełniają rekurencję*

$$\left[\begin{array}{c} n+1 \\ k \end{array} \right] = \left[\begin{array}{c} n \\ k-1 \end{array} \right] + n \left[\begin{array}{c} n \\ k \end{array} \right], \quad 1 \leq k \leq n-1$$

przy czym

$$\left[\begin{array}{c} 0 \\ 0 \end{array} \right] = 1, \quad \left[\begin{array}{c} n \\ 0 \end{array} \right] = 0, \quad n > 0.$$

Dowód: Założmy, że mamy $n+1$ obiektów w k cyklach. Ta sytuacja mogła powstać na dwa sposoby: albo do permutacji n obiektów o $k-1$ cyklach dodaliśmy jeden obiekt, tworząc nowy cykl singletowy, co prowadzi do $\left[\begin{array}{c} n \\ k-1 \end{array} \right]$ możliwości z wcześniejszej permutacji, albo do permutacji n obiektów o k cyklach włączliśmy nowy element do któregoś z istniejących już cykli. Niech te

¹ Używamy tu tzw. *bezznakowych* liczb Stirlinga pierwszego rodzaju. Niektórzy autorzy dodają do definicji kombinatorycznej znak $(-1)^{n-k}$. Nasza notacja dla liczb Stirlinga pierwszego i drugiego rodzaju jest zgodna z [3].

			1					
		0		1				
	0		1		1			
	0	2		3		1		
	0	6	11	6		1		
	0	24	50	35	10		1	
	0	120	274	225	85	15		1
0	720	1764	1624	735	175	21		1

Rysunek 4.3: Trójkąt liczb Stirlinga pierwszego rodzaju, utworzony zgodnie z rekurencją 4.1. Wartość liczby obiektów n uzyskujemy, licząc kolejne wiersze od góry, zaczynając liczenie od 0, a wartość k dostajemy licząc kolejne kolumny, zaczynając od lewej strony i również zaczynając liczenie od 0. Na przykład dla $n = 4$ i $k = 2$ uzyskujemy 11

cykle mają długosci j_1, j_2, \dots, j_k , z oczywistym warunkiem $j_1 + j_2 + \dots + j_k = n$. W każdym cyklu nowy element można dołączyć na j_m sposobów, zatem łączna liczba możliwości dołączenia nowego elementu jest równa n razy liczba istniejących już permutacji n obiektów o k cyklach, co daje żądaną rekurencję. Warunek $\begin{bmatrix} n \\ 0 \end{bmatrix} = 0$ wynika z faktu, że każda permutacja daje się rozłożyć na cykle i nie-możliwa jest sytuacja $k = 0$. Równość $\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1$ pozwala stosować rekurencję dla $n = 0$ i $k = 1$. Mamy wówczas

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1 + 0,$$

co daje oczywisty wynik $\begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1$. \square

Trójkąt liczb Stirlinga pierwszego rodzaju ukazany jest na rys. 4.3.

Problem 4.2 (grupy dzieci). Po skończonych płasach n przedszkolaków zostało przez studenta podzielonych na k grup. Na ile sposobów można to zrobić?

Mówiąc nieco inaczej, pytamy na ile sposobów można włożyć n rozróżnialnych piłek do k nierożnialnych wiaderek tak, aby żadne wiaderko nie pozostało puste. Na przykład cztery rozróżnialne piłki o numerach 1, 2, 3 i 4 możemy włożyć do trzech nierożnialnych wiaderek na następujących 6 sposobów:

$$\begin{aligned} \{1, 2\} \cup \{3\} \cup \{4\}, & \quad \{1, 3\} \cup \{2\} \cup \{4\}, & \quad \{1, 4\} \cup \{2\} \cup \{3\}, \\ \{2, 3\} \cup \{1\} \cup \{4\}, & \quad \{2, 4\} \cup \{1\} \cup \{3\}, & \quad \{3, 4\} \cup \{1\} \cup \{2\}. \end{aligned}$$

Def. 4.2. Podziałem n -elementowego zbioru X na k (niepustych parami rozłącznych) podzbiorów nazywamy rodzinę zbiorów $\{A_1, A_2, \dots, A_k\}$ spełniającą warunki $A_i \neq \emptyset$, $A_i \subset X$, $A_i \cap A_j = \emptyset$ dla $i \neq j$ oraz $A_1 \cup A_2 \cup \dots \cup A_k = X$.

Def. 4.3. Liczby Stirlinga drugiego rodzaju,

$$\left\{ \begin{array}{c} n \\ k \end{array} \right\},$$

to liczba podziałów n -elementowego zbioru na k (niepustych parami rozłącznych) podzbiorów.

Tw. 4.2. Liczby Stirlinga drugiego rodzaju spełniają rekurencję

$$\left\{ \begin{array}{c} n+1 \\ k \end{array} \right\} = k \left\{ \begin{array}{c} n \\ k \end{array} \right\} + \left\{ \begin{array}{c} n \\ k-1 \end{array} \right\}, \quad 1 \leq k \leq n-1,$$

przy czym

$$\left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\} = 1, \quad \left\{ \begin{array}{c} n \\ 0 \end{array} \right\} = 0, \quad n > 0.$$

Dowód: Mając n obiektów podzielonych na k podzbiorów, możemy $n+1$ obiekt dołączyć do któregoś z istniejących podzbiorów, co daje $k \left\{ \begin{array}{c} n \\ k \end{array} \right\}$ możliwości, lub też do n obiektów podzielonych na $k-1$ podzbiorów możemy dołączyć nowy jednoelementowy podzbiór, co daje $\left\{ \begin{array}{c} n \\ k-1 \end{array} \right\}$ możliwości. \square

Szczególne wartości liczb Stirlinga są następujące:

$$\left[\begin{array}{c} n \\ 1 \end{array} \right] = (n-1)!, \tag{4.2}$$

$$\left\{ \begin{array}{c} n \\ 1 \end{array} \right\} = 1, \quad n \geq 1, \tag{4.3}$$

$$\left[\begin{array}{c} n \\ n \end{array} \right] = \left\{ \begin{array}{c} n \\ n \end{array} \right\} = 1, \tag{4.4}$$

$$\left[\begin{array}{c} n \\ n-1 \end{array} \right] = \left\{ \begin{array}{c} n \\ n-1 \end{array} \right\} = \binom{n}{2} = \frac{n(n-1)}{2}. \tag{4.5}$$

Własności te wynikają w prosty sposób z interpretacji kombinatorycznej i ich dowód pozostawiony jest jako ćw. 1. Trójkąt liczb Stirlinga drugiego rodzaju przedstawiony jest na rys. 4.4.

			1					
		0		1				
		0		1		1		
		0		1		3		1
		0		1		7		6
		0		1		15		10
		0		1		31		15
		0		1		301		21
		0		1		63		1
						350		
						140		
						90		
						65		
						1		

Rysunek 4.4: Trójkąt liczb Stirlinga drugiego rodzaju, utworzony zgodnie z rekurencją 4.2. Wartości n i k znajdujemy tak samo, jak na rys. 4.3

Liczby Stirlinga drugiego rodzaju są zatem związane z problemem kombinatorycznym liczby różnych funkcji z n -elementowego zbioru A na k -elementowy zbiór B . Specyfikacja na a nie w jest tu bardzo istotna, w przypadku funkcji w mamy bowiem wariacje z powtóreniami omówione szczegółowo w podrozdziale 2.3. Specyfikacja na wprowadza ograniczenie: żaden element zbioru B nie może zostać nieprzyporządkowany. Chwila zastanowienia i stwierdzamy, że

Tw. 4.3. *Liczba różnych funkcji ze zbioru n -elementowego na zbiór k -elementowy wynosi $k! \left\{ \begin{array}{c} n \\ k \end{array} \right\}$.*

Dowód: Dzielimy n elementowy zbiór A na k podzbiorów, co daje $\left\{ \begin{array}{c} n \\ k \end{array} \right\}$ możliwości. Wszystkie elementy z danego podzbioru przyporządkujemy temu samemu elementowi z k -elementowego zbioru B . Elementom z pierwszego podzbioru zbioru A możemy przyporządkować jeden z k elementów zbioru B , elementom z drugiego podzbioru zbioru A ($k - 1$) pozostałych elementów zbioru B , z trzeciego podzbioru ($k - 2$) elementów itd., co łącznie daje $k!$ możliwości dla każdego podziału zbioru A . \square

4.3 Liczby Bella

Z liczbami Stirlinga drugiego rodzaju blisko związane są liczby Bella².

² Eric Temple Bell (1883-1960), szkocki matematyk mieszkający w USA, pisał również książki science fiction.

Def. 4.4. *Liczba podziałów zbioru n -elementowego na dowolną liczbę (niepustych parami rozłącznych) podzbiorów nazywa się liczbą Bella B_n .*

Pytamy więc, na ile sposobów możemy podzielić na grupy naszych n przedszkolaków, bez specyfikowania, ile tych grup ma być. Z definicji liczb Stirlinga drugiego rodzaju wynika natychmiast, że

$$B_n = \sum_{k=0}^n \left\{ \begin{array}{c} n \\ k \end{array} \right\}. \quad (4.6)$$

Kilka pierwszych wartości liczb Bella to, poczynając od $n = 0$,

$$1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, \dots$$

Tw. 4.4. *Liczby Bella spełniają rekurencję*

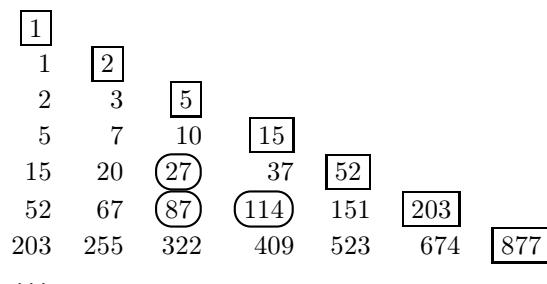
$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k. \quad (4.7)$$

Dowód: Założymy, że mamy zbiór n elementów. Wybieramy zeń k elementów, co możemy zrobić na $\binom{n}{k}$ sposobów, i dołączamy do tego podzbioru $n + 1$ element. Pozostałe $n - k$ elementów możemy podzielić, z definicji liczby Bella, na B_{n-k} sposobów. Sumując po k od 0 do n , otrzymujemy

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_{n-k} = \sum_{l=0}^n \binom{n}{n-l} B_l = \sum_{l=0}^n \binom{n}{l} B_l, \quad (4.8)$$

gdzie $l = n - k$ oraz skorzystaliśmy z (2.11). \square

Liczby Bella można uzyskać z *trójkąta Bella* (rys. 4.5) skonstruowanego w następujący sposób: pierwszy wiersz zawiera 1 (oraz zera, których nie wypisujemy),



Rysunek 4.5: Trójkąt Bella. Kolejne liczby Bella, począwszy od $B_1 = 1$, zaznaczono prostokątnymi ramkami. Owale wskazują konstrukcję, np. $114 = 87 + 27$

podobnie jak w innych trójkątach tego typu). Wiersz numer n jest tworzony w następujący sposób: pierwszy wyraz po lewej stronie, który piszemy pod pierwszym wyrazem wiersza $n - 1$, jest przepisany ostatnim wyrazem po prawej stronie z wiersza $n - 1$, a kolejne wyrazy są sumą wyrazu sąsiedniego po lewej stronie z tego samego wiersza i wyrazu nad tym wyrazem, np. $114 = 87 + 27$. Wiersz numer n zawiera n wyrazów. Liczby Bella B_1, B_2, \dots , odczytujemy z ostatnich wyrazów po prawej stronie każdego wiersza.

Koniec tego podrozdziału stanowi bardziej zaawansowany materiał, ukazujący pewne typowe metody analityczne pomocne w rozważaniach dotyczących obiektów kombinatorycznych. Aby móc więcej powiedzieć o liczbach Bella, skorzystamy z *wykładniczej funkcji tworzącej*.

Def. 4.5. Wykładniczą funkcją tworzącą dla ciągu (a_n) , $n \geq 0$, jest szereg

$$g(z) = \sum_{n=0}^{\infty} \frac{a_n}{n!} z^n.$$

Różnica ze zwykłą funkcją tworzącą z def. 1.9 tkwi tutaj w podzieleniu sumowanych wyrazów przez $n!$. Decyzja, czy użyć zwykłej, czy wykładniczej funkcji tworzącej zależy od rozważanego problemu – czasem lepsze efekty daje funkcja zwykła, czasem wykładnicza.

Rozważmy ponownie rekurencję dla liczb Stirlinga drugiego rodzaju, Tw. 4.1. Mnożąc obie strony rekurencji przez $x^n/n!$ i sumując po n począwszy od $n = k - 1$, uzyskujemy związek

$$\sum_{n=k-1}^{\infty} \left\{ \begin{array}{c} n+1 \\ k \end{array} \right\} \frac{x^n}{n!} = k \sum_{n=k}^{\infty} \left\{ \begin{array}{c} n \\ k \end{array} \right\} \frac{x^n}{n!} + \sum_{n=k-1}^{\infty} \left\{ \begin{array}{c} n \\ k-1 \end{array} \right\} \frac{x^n}{n!}, \quad (4.9)$$

gdzie w pierwszym członie po prawej stronie mogliśmy zmienić dolną granicę sumowania, ponieważ $\left\{ \begin{array}{c} k-1 \\ k \end{array} \right\} = 0$. Utwórzmy teraz wykładniczą funkcję tworzącą dla ustalonego k w następujący sposób:

$$g_k(x) = \sum_{n=k}^{\infty} \left\{ \begin{array}{c} n \\ k \end{array} \right\} \frac{x^n}{n!}. \quad (4.10)$$

Wówczas

$$\frac{d}{dx} g_k(x) = \sum_{n=k}^{\infty} \left\{ \begin{array}{c} n \\ k \end{array} \right\} \frac{x^{n-1}}{(n-1)!} = \sum_{n=k-1}^{\infty} \left\{ \begin{array}{c} n+1 \\ k \end{array} \right\} \frac{x^n}{n!},$$

zatem równanie (4.9) można zapisać jako rekurencyjny ciąg równań różniczkowych

$$g'_k(x) = kg_k(x) + g_{k-1}(x), \quad k \geq 1, \quad (4.11)$$

przy czym $g_0(x) = \begin{Bmatrix} n \\ k \end{Bmatrix} = 1$. Łatwo sprawdzić poprzez różniczkowanie, że rozwiązanie stanowią funkcje

$$g_k(x) = \frac{1}{k!}(e^x - 1)^k. \quad (4.12)$$

Wykładnicza funkcja tworząca dla liczb Bella, będących sumą liczb Stirlinga drugiego rodzaju (4.6), jest sumą funkcji (4.12). Możemy tu rozciągnąć sumę we wskaźniku k do nieskończoności, ponieważ $\begin{Bmatrix} n \\ k \end{Bmatrix} = 0$ dla $k > n$. Mamy zatem

$$G(x) = \sum_{j=0}^{\infty} B_j \frac{x^j}{j!} = \sum_{k=0}^{\infty} g_k(x) = e^{e^x - 1} = \frac{1}{e} \sum_{j=0}^{\infty} \frac{e^{jx}}{j!}.$$

Różniczkując po x , otrzymujemy

$$B_n = \left. \frac{d^n G(x)}{dx^n} \right|_{x=0} = \frac{1}{e} \sum_{j=0}^{\infty} \frac{j^n}{j!},$$

co jest tzw. *wzorem Dobińskiego*.

4.4 Triangulacja wielokąta i liczby Catalana

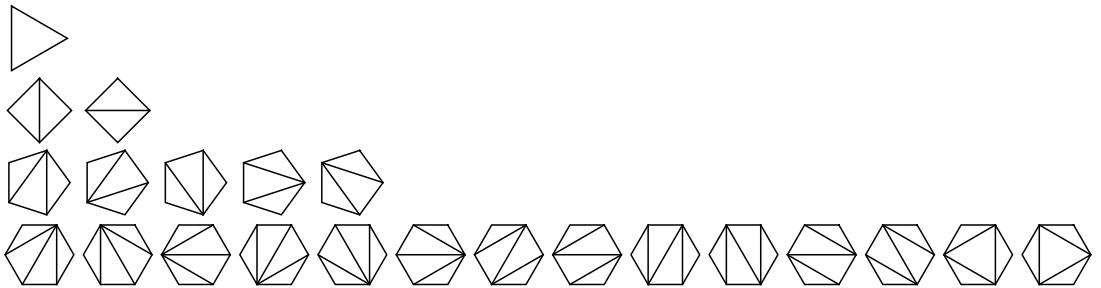
Rozważmy teraz następujący problem prowadzący do tzw. liczb Catalana³:

Problem 4.3 (triangulacja wielokąta). *Na ile sposobów można przeprowadzić triangulację (podział na trójkąty) wielokąta wypukłego o $n + 2$ wierzchołkach?*

Zaczynamy pieczołowicie rysować te wielokąty, począwszy od $n = 1$ (trójkąta), następnie $n = 2$ (czworokąta) itd., co przedstawiono na rys. 4.6. Widzimy, że liczba możliwości, którą oznaczamy jako C_n , rośnie szybko z n : 1, 2, 5, 13, ... Jakie jest ogólne rozwiązanie? Aby napisać stosowną rekurencję, rozważmy konstrukcję z rys. 4.7, przedstawiającą podział dziewięciokąta, a więc $n = 7$. Skupmy uwagę na jednym z wierzchołków. Wierzchołek ten należy do wierzchołka jednego z trójkątów, oznaczonego szarym odcieniem. Bok przeciwległy temu wierzchołkowi jest jednym z boków dziewięciokąta, mamy więc 7 możliwości zobrazowanych na rys. 4.7. Teraz następuje kluczowa obserwacja. Po obu stronach trójkąta znajdują się wielokąty, które możemy dalej dzielić rekurencyjnie. Na przykład na prawej górnej części rysunku po lewej stronie trójkąta znajduje się czworokąt ($n = 2$), a po prawej sześciokąt ($n = 4$). Mamy zatem równanie

$$C_7 = C_0 C_6 + C_1 C_5 + C_2 C_4 + C_3 C_3 + C_4 C_2 + C_5 C_1 + C_6 C_0, \quad (4.13)$$

³ Eugène Charles Catalan (1814-1894), belgijski matematyk.

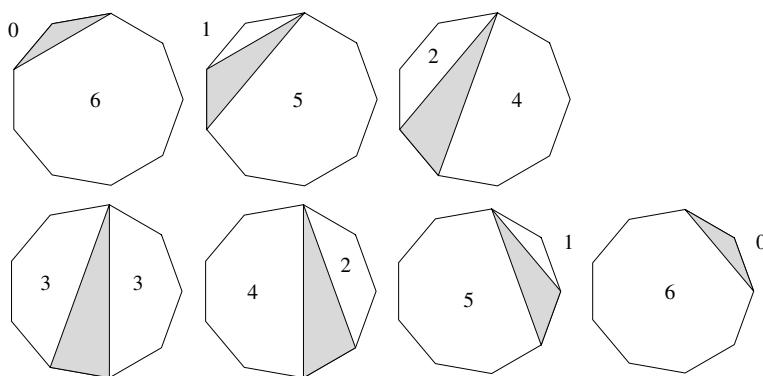
Rysunek 4.6: Triangulacje wielokątów o liczbie wierzchołków $n + 2 = 3, 4, 5$ i 6

w którym kolejne człony po prawej stronie odpowiadają odpowiednim częściom rys. 4.7. Skrajne przypadki mają tylko jeden wielokąt oprócz trójkąta, więc musimy przyjąć $C_0 = 1$, aby zachować poprawne zliczanie. Uzyskaliśmy więc wyrażenie na C_7 w postaci sum iloczynów dwóch liczb C_i , gdzie suma wskaźników w każdym członie wynosi 6. Uogólniając na dowolne n , w prosty sposób uzyskujemy związek

$$C_{n+1} = C_0 C_n + C_1 C_{n-1} + C_2 C_{n-2} + \cdots + C_n C_0 = \sum_{k=0}^n C_k C_{n-k}. \quad (4.14)$$

Aby rozwiązać rekurencję (4.14), skorzystamy z techniki opartej na funkcjach tworzących dyskutowanych w rozdz. 1.4. W tym celu mnożymy obie strony równania (4.14) przez x^n i sumujemy po n :

$$\sum_{n=0}^{\infty} C_{n+1} x^n = \sum_{n=0}^{\infty} \sum_{k=0}^n C_k x^k C_{n-k} x^{n-k}.$$



Rysunek 4.7: Konstrukcja rekurencji 4.13

Oznaczając $c(x) = \sum_{n=0}^{\infty} C_n x^n$, po lewej stronie powyższej równości mamy

$$\sum_{m=1}^{\infty} C_m x^{m-1} = \frac{1}{x} \left[\sum_{m=0}^{\infty} C_m x^m - C_0 \right] = \frac{1}{x} [c(x) - C_0] = \frac{1}{x} [c(x) - 1],$$

a po prawej stronie rozpoznajemy iloczyn Cauchy'ego szeregów

$$\sum_{n=0}^{\infty} \sum_{k=0}^n C_k x^k C_{n-k} x^{n-k} = \sum_{m=0}^{\infty} C_m x^m \sum_{k=0}^{\infty} C_k x^k = c(x)^2.$$

Otrzymujemy zatem równanie kwadratowe na $c(x)$ postaci

$$xc(x)^2 = c(x) - 1,$$

o rozwiązaniach $c(x) = (1 \pm \sqrt{1-4x})/(2x)$. Aby rozstrzygnąć, które rozwiązanie jest właściwe, korzystamy z faktu $c(0) = C_0 = 1$. Rozwiązanie z plusem jest osobliwe, ponieważ $\lim_{x \rightarrow 0} (1 + \sqrt{1-4x})/(2x) = \infty$. Rozwiązanie z minusem daje poprawny wynik, ponieważ

$$\lim_{x \rightarrow 0} \frac{1 - \sqrt{1-4x}}{2x} = \lim_{x \rightarrow 0} \frac{(1 - \sqrt{1-4x})(1 + \sqrt{1-4x})}{2x(1 + \sqrt{1-4x})} = \lim_{x \rightarrow 0} \frac{4x}{2x(1 + \sqrt{1-4x})} = 1,$$

zatem

$$c(x) = \frac{1 - \sqrt{1-4x}}{2x}.$$

Następnie rozwijamy $\sqrt{1-4x}$ za pomocą wzoru dwumianowego (2.17), dostając

$$\begin{aligned} c(x) &= \frac{1}{2x} \left(1 - \sum_{n=0}^{\infty} \binom{\frac{1}{2}}{n} (-4x)^n \right) = -\frac{1}{2x} \sum_{n=1}^{\infty} \binom{\frac{1}{2}}{n} (-4x)^n \\ &= 2 \sum_{m=0}^{\infty} \binom{\frac{1}{2}}{m+1} (-4x)^m, \end{aligned}$$

gdzie $m = n - 1$, po czym pracowicie przekształcamy współczynnik przy x^m :

$$\begin{aligned} 2 \binom{\frac{1}{2}}{m+1} (-4)^m &= 2 \frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2)\dots(\frac{1}{2}-m)}{(m+1)!} (-4)^m \\ &= \frac{(2-1)(4-1)\dots(2m-1)}{(m+1)!} 2^m = \frac{1 \cdot 3 \cdot \dots \cdot (2m-1)}{m!(m+1)} 2^m \\ &= \frac{1 \cdot 3 \cdot \dots \cdot (2m-1)}{(m+1)m!} \frac{2 \cdot 4 \cdot \dots \cdot 2m}{m!} = \frac{(2m)!}{(m+1)m!^2} = \frac{1}{m+1} \binom{2m}{m}. \end{aligned}$$

Tak więc

$$c(x) = \sum_{n=0}^{\infty} \frac{1}{n+1} \binom{2n}{n} x^n,$$

skąd odczytujemy wzór ogólny na liczby Catalana:

$$C_n = \frac{1}{n+1} \binom{2n}{n}. \quad (4.15)$$

Liczby Catalana występują w wielu zagadnieniach kombinatorycznych (zob. ćw. 4.6).

4.5 Partycje liczb

W tym i następnym podrozdziale zajmiemy się problemem rozkładu (czyli *partycji*) liczby naturalnej na składniki, znanym jako *partitio numerorum*. Interesuje nas, na ile różnych sposobów można przedstawić liczbę n w postaci liczb należących do pewnego podzbioru liczb naturalnych U . Podzióbior U zależy od konkretnego problemu.

Weźmy na początek problem rozkładu liczby n na składniki naturalne, np. liczbę 6 możemy rozłożyć na następujące 11 sposobów:

$$\begin{aligned} 6 &= 1 + 1 + 1 + 1 + 1 + 1 \\ 6 &= 2 + 1 + 1 + 1 + 1 \\ 6 &= 2 + 2 + 1 + 1 \\ 6 &= 2 + 2 + 2 \\ 6 &= 3 + 1 + 1 + 1 \\ 6 &= 3 + 2 + 1 \\ 6 &= 3 + 3 \\ 6 &= 4 + 1 + 1 \\ 6 &= 4 + 2 \\ 6 &= 5 + 1 \\ 6 &= 6 \end{aligned} \quad (4.16)$$

Liczbę partycji liczby n oznaczamy jako $p(n)$. Zwarta postać $p(n)$ nie jest znana, a teoria rozkładów stanowi jedną z ważnych dziedzin teorii liczb. Problem znalezienia $p(n)$ można za pomocą techniki funkcji tworzących sprowadzić do równoważnego problemu znajdowania współczynników przy danym wyrazie iloczynu wielomianów. Podstawową sztuczką jest tu zastosowanie oczywistej własności potęgowania $z^a z^b \dots z^m = z^{a+b+\dots+m}$, gdzie $n = a + b + \dots + m$. Rozważmy następujące wyrażenie:

$$\begin{aligned} f(z) &= (1 + z + z^2 + z^3 + z^4 + z^5 + z^6 + \dots)(1 + z^2 + (z^2)^2 + (z^2)^3 + \dots) \cdot \\ &\quad (1 + z^3 + (z^3)^2 + \dots)(1 + z^4 + (z^4)^2 + \dots)(1 + z^5 + \dots)(1 + z^6 + \dots) \dots \end{aligned} \quad (4.17)$$

Pierwszy wielomian oznacza „weź zero jedynek (człon $1 = z^0$) lub weź jedną jedynkę (człon $z = z^1$) lub weź dwie jedynki (człon z^2)” itd., drugi wielomian oznacza „weź zero dwójkę (człon $1 = z^0$) lub weź jedną dwójkę (człon z^2) lub weź dwie dwójkę (człon $(z^2)^2 = z^4$)” itd. Wyobraźmy sobie dla ustalenia uwagi, że wymnażamy te wielomiany i szukamy współczynnika przy potędze z^6 . Mamy tu kilka możliwości: możemy wziąć z^6 z pierwszego wielomianu oraz same jedynki z pozostałych, co oznacza wzięcie sześciu jedynek, czyli partycji $1+1+1+1+1+1$, możemy też wziąć np. z^3 z pierwszego wielomianu i z^3 z trzeciego wielomianu, co odpowiada partycji $1+1+1+3$. Wymnożenie wielomianów i znalezienie współczynnika przy potędze z^6 „automatycznie” policzy nam liczbę partycji $p(6)$! Oczywiście wymnażanie wielomianów jest czynnością pracochlonną, ktoś mógłby więc powiedzieć, że jak dotąd niewiele zyskaliśmy. Jednak zamieniliśmy problem kombinatoryczny na zagadnienie mnożenia wielomianów. Wynik tego mnożenia jest następujący

$$f(z) = \sum_{n=0}^{\infty} p_n z^n = 1 + z + 2z^2 + 3z^3 + 5z^4 + 7z^5 + 11z^6 + \dots \quad (4.18)$$

Rzeczywiście, współczynnik przy z^6 wynosi 11, czyli tyle, ile doliczyliśmy się „na piechotę” w rozkładach (4.16).

Kropki w wyrażeniu 4.17 oznaczają dalsze potęgi z , które w rozkładzie liczby 6 nie miały znaczenia, bo dawały większą potęgę. Możemy więc te sumowania rozciągnąć do nieskończoności i użyć wzoru $\sum_{k=0}^{\infty} a^k = 1/(1-a)$, co daje

$$f(z) = \frac{1}{1-z} \frac{1}{1-z^2} \frac{1}{1-z^3} \frac{1}{1-z^4} \frac{1}{1-z^5} \frac{1}{1-z^6} \dots, \quad (4.19)$$

gdzie \dots oznaczają wyższe człony, niezbędne dopiero przy szukaniu partycji dla $n > 6$. Następnie musimy rozwinąć $f(z)$ w szereg Taylora wokół $z = 0$ i gotowe. Dla większych n najlepiej zastosować program manipulacji symbolicznych, jak *Mathematica*, *Maple*, *Form* itp. W ten sposób (za pomocą komputera) możemy łatwo uzyskać wartości $p(n)$ dla początkowych n , zaczynając od $n = 1$:

$$1, 2, 3, 5, 7, 11, 15, 22, 30, 42, 56, 77, 101, 135, 176, 231, 297, 385, 490, 627, 792, \dots$$

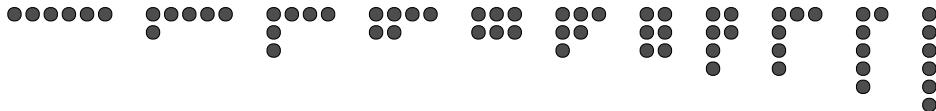
Dla bardzo dużych n problem jest trudny.

Widzimy, że liczba partycji $p(n)$ bardzo szybko rośnie z n . Hardy⁴ i Ramanujan⁵ pokazali, że dla dużych n wartość $p(n)$ można oszacować wzorem asymptotycznym [7]

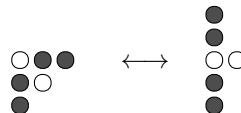
$$p(n) \sim \frac{1}{4\sqrt{3}n} e^{\pi\sqrt{2n/3}}.$$

⁴ Godfrey Harold Hardy (1877-1947), brytyjski matematyk, zasłynął z osiągnięć w teorii liczb i analizie matematycznej. Jak twierdził, jego największe odkrycie to znalezienie Ramanujana.

⁵ Srinivasa Aiyangar Ramanujan (1887-1920), genialny indyjski matematyk, bardzo istotnie przyczynił się i niezwykle zainspirował rozwój teorii liczb, funkcji modularnych i teorii partycji.



Rysunek 4.8: Diagramy Ferrersa dla rozkładów liczby 6 (4.16). Diagram odpowiadający podziałowi $3 + 2 + 1$ jest samosprzężony



Rysunek 4.9: Graficzny dowód Tw. 4.5. Każdy diagram samosprzężony można rozwiniąć jak na rysunku, uzyskując diagram rozkładu na liczby nieparzyste wzajemnie różne, w tym przypadku 5 i 1. Odwrotnie, każdy diagram rozkładu na różne składniki nieparzyste można zwinąć do diagramu samosprzężonego

Partycje wygodnie jest reprezentować graficznie w postaci tzw. *diagramów Ferrersa*⁶. Na przykład partycje liczby 6 wymienione w (4.16) możemy przestawić jako diagramy przedstawione na rys. 4.8. Suma kropek w każdym wierszu diagramu Ferrersa daje składnik partycji, np. pierwszy diagram na rys. 4.8 to $1 + 1 + 1 + 1 + 1 + 1$, drugi to $2 + 1 + 1 + 1 + 1$ itd. Diagramy uzyskane przez zamianę wierszy z kolumnami nazywamy *dualnymi* lub *sprzężonymi*. Z konstrukcji diagramów Ferrersa wynika, że jeśli diagram wyjściowy ma k składników, to diagram sprzężony ma największy składnik równy k . Na przykład na rys. 4.8 trzeci diagram od lewej, $2 + 2 + 1 + 1$, ma 4 składniki, a sprzężony do niego diagram trzeci od prawej, $4 + 2$, ma największy składnik 4. Widzimy, że pary diagramów o numerach, licząc od lewej, 1-11, 2-10, 3-9, 4-8 i 5-7 są sprzężone. Diagram 6, odpowiadający rozkładowi $3 + 2 + 1$, jest *samosprzężony*, czyli równy otrzymanemu zeń diagramowi sprzężonemu.

Za pomocą diagramów Ferrersa można bardzo łatwo udowodnić kilka niewątpliwie prawdziwych twierdzeń o rozkładach, stąd diagramy te są użyteczne.

Tw. 4.5. *Liczba rozkładów samosprzężonych liczby n jest równa liczbie rozkładów n na różne składniki nieparzyste.*

Dowód: Idea dowodu przedstawiona jest graficznie na rys. 4.9. W diagramie samosprzężonym „rozprostowujemy rzędy paciorków”, tak jak na rysunku, i uzyskujemy diagram o nieparzystej liczbie paciorków w każdej kolumnie, ponadto liczby te są różne w każdej kolumnie. W ten sposób powstaje odpowiedniość 1-1 między diagramami samosprzężonymi i diagramami rozkładów na różne składniki nieparzyste. \square

Zajmijmy się teraz liczbą partycji n na dokładnie k składników, oznaczoną jako $p(n, k)$. Na przykład $p(6, 2) = 3$, patrz rys. 4.8.

⁶ Norman Macleod Ferrers (1829-1903), matematyk brytyjski.

Tw. 4.6. *Liczba rozkładów liczby n na k składników, $p(n, k)$, jest równa liczbie rozkładów, gdzie k jest największym składnikiem.*

Dowód: Twierdzenie wynika natychmiast z faktu, że sprzężenie zamieniające wiersze i kolumny w diagramie Ferrersa odpowiada zamianie liczby składników w diagramie wyjściowym na największy składnik w diagramie sprzężonym. \square

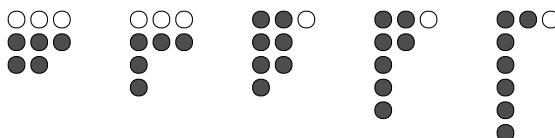
A teraz istotne twierdzenie o zależności rekurencyjnej liczb $p(n, k)$, które pozwoli w łatwy sposób konstruować liczbę partycji.

Tw. 4.7. *Partycje liczby n na k składników spełniają rekurencję*

$$p(n, k) = p(n - 1, k - 1) + p(n - k, k). \quad (4.20)$$

Zanim przystąpimy do dowodu, rozważmy przykład partycji liczby 8 na 3 składniki, ukazany na rys. 4.10. Widzimy, że wszystkie możliwe diagramy rozpadają się na dwie klasy: pierwszą, z taką samą liczbą paciorków w dwóch górnych rzędach (dwa pierwsze diagramy od lewej), oraz drugą, z liczbą paciorków w górnym rzędzie większą od liczby paciorków w drugim rzędzie od góry (trzy diagramy od prawej strony). Możemy wyróżnić dla wygody odpowiednie paciorki kolorem białym. W pierwszej klasie diagramów są to paciorki z górnego rzędu, których we wszystkich diagramach klasy pierwszej jest tyle samo, tutaj $k = 3$. A zatem liczba podziałów w tej klasie jest równa liczbie podziałów czarnych paciorków, których jest $n - k = 8 - 3 = 5$. A zatem liczba podziałów w klasie pierwszej wynosi $p(8 - 3, 3)$. W drugiej klasie diagramów, gdzie biały koralik dołączony jest na końcu pierwszego rzędu, podziałów jest tyle, co podziałów $n - 1 = 7$ czarnych koralików, ale tym razem na $k - 1 = 2$ składniki. Liczba możliwości to $p(7, 2)$. Mamy więc $p(8, 3) = p(7, 2) + p(8 - 3, 3)$.

Dowód: Dowód w ogólnym przypadku polega na idei zliczania pokazanej na rys. 4.10, tyle, że identyfikujemy $8 \rightarrow n$, $3 \rightarrow k$. Biały paciorków w pierwszej klasie diagramów jest n , czarnych $n - k$, długość górnego wiersza z czarnych



Rysunek 4.10: Diagramy Ferrersa dla rozkładu liczby 8 na trzy składniki. Diagramy rozpadają się na dwie klasy: z taką samą liczbą paciorków w dwóch górnych rzędach (dwa pierwsze diagramy od lewej), oraz z liczbą paciorków w górnym rzędzie większą od liczby paciorków w drugim rzędzie od góry (trzy diagramy od prawej strony). Wyróżniając dla wygody stosowne paciorki kolorem białym, widzimy, że w pierwszej klasie diagramów liczba podziałów odpowiada podziałom czarnych paciorków, czyli $p(8 - 3, 3)$. W drugiej klasie diagramów jedyny biały koralik dołączony jest na końcu pierwszego rzędu. Podziałów wśród czarnych koralików jest $p(7, 2)$. Mamy więc $p(8, 3) = p(7, 2) + p(5, 3)$.

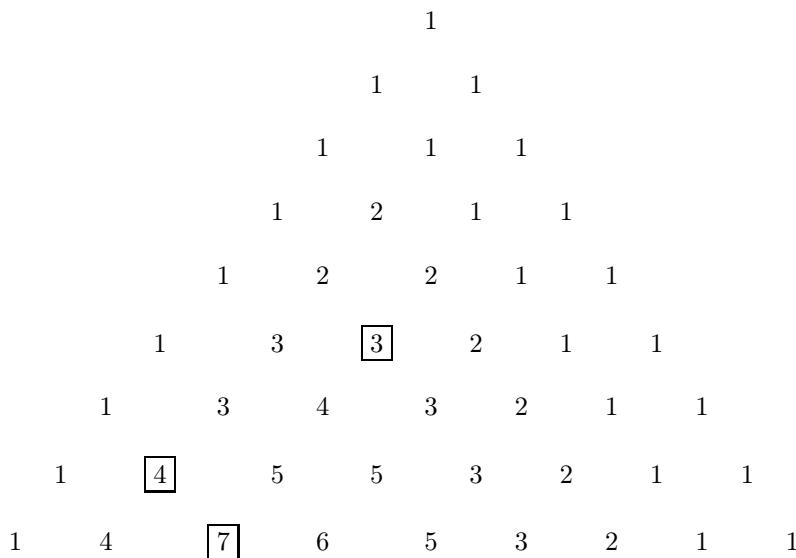
paciorków wynosi k , jest więc $p(n-k, k)$ możliwości podziałów. W drugiej klasie jest jeden biały paciorek i $n-1$ czarnych, a długość górnego wiersza z czarnych paciorków wynosi $k-1$, więc mamy w tym przypadku $p(n-1, k-1)$ możliwości. Dodanie możliwości z obu klas diagramów daje podaną rekurencję. \square

Zauważmy, że rekurencja z Tw. 4.7 sięga k kroków wstecz, co odróżnia ją od poznanych wcześniej przypadków dla symbolu Newtona czy liczb Stirlinga. Trójkąt dla liczb $p(n, k)$, skonstruowany zgodnie z wzorem (4.7), przedstawiony jest na rys. 4.11. Oczywiście, suma wyrazów w każdym wierszu daje liczbę $p(n)$ wszystkich partycji liczby n . Zauważmy też, badając bacznie trójkąt z rys. 4.11 (zob. ēw. 10), że dla $k=2$ i 3 mamy związki

$$\begin{aligned} p(n, 2) &= \lfloor n/2 \rfloor, \\ p(n, 3) &= \lfloor n^2/12 \rfloor, \end{aligned} \quad (4.21)$$

gdzie funkcję najbliższej liczby całkowitej $\lfloor \cdot \rfloor$ zdefiniowano w podrozdziale 1.8.

Zajmijmy się teraz partycjami na składniki *różne*, tzn. składniki niemogące się powtarzać. Liczbę tych partycji dla liczby n oznaczamy jako $q(n)$. Na przykład



Rysunek 4.11: Trójkąt liczby partycji n na k składników, $p(n, k)$, gdzie $n = 1, 2, \dots$ numeruje wiersze, a $k = 1, 2, \dots, n$ kolumny. Rekurencja 4.7 prowadzi do następującej konstrukcji: dana liczba jest sumą liczby bezpośrednio powyżej na linii idącej w górę w lewo oraz liczby na linii idącej w górę w prawo, ale odsuniętej o odległość danej liczby od lewego brzegu trójkąta. Na przykład liczby w ramkach to $p(9, 3) = 7$, $p(8, 2) = 4$, $p(6, 3) = 3$, $7 = 4 + 3$. Suma wyrazów w każdym wierszu daje całkowitą liczbę podziałów liczby n , $p(n)$.

liczbę 9 możemy rozłożyć z tym ograniczeniem na następujące sposoby:

$$\begin{aligned}
 9 &= 9 \\
 9 &= 8 + 1 \\
 9 &= 7 + 2 \\
 9 &= 6 + 3 \\
 9 &= 6 + 2 + 1 \\
 9 &= 5 + 4 \\
 9 &= 5 + 3 + 1 \\
 9 &= 4 + 3 + 2
 \end{aligned} \tag{4.22}$$

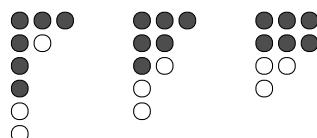
czyli $q(9) = 8$. Analogicznie do notacji dla zwykłych partycji, $q(n, k)$ oznacza liczbę partycji n na k różnych składników. Diagramy Ferrersa dla rozkładu liczby 9 na trzy składniki różne przedstawione są na rys. 4.12.

Przedstawimy teraz ciekawy związek:

Tw. 4.8. *Liczby partycji na k składników dowolnych i składników różnych spełniają związek*

$$q(n, k) = p\left(n - \frac{k(k-1)}{2}, k\right). \tag{4.23}$$

Dowód: Prześledźmy najpierw przykład zilustrowany diagramami Ferrersa na rys. 4.12. Partycje liczby 9 na 3 składniki różne powstały ze zwykłych partycji liczby 6 na 3 składniki poprzez dodanie dwóch jasnych kropek do pierwszego składnika i jednej jasnej kropki do drugiego składnika. W ogólności dodalibyśmy $(k-1)$ kropek do pierwszego składnika, $(k-2)$ do drugiego, ..., wreszcie jedną do składnika na pozycji $(k-1)$. Łącznie daje to $k(k-1)/2$ dodanych kropek. Ponieważ, idąc teraz od prawej do lewej strony, dodajemy w każdym składniku o jedną kropkę więcej, składniki, które są równe w rozkładzie liczby $n - k(k-1)/2$, generują różne składniki liczby n . Tak więc liczba partycji liczby n na k składników różnych jest równa liczbie partycji liczby $n - k(k-1)/2$ na k składników, które mogą się powtarzać. \square



Rysunek 4.12: Diagramy Ferrersa dla rozkładu liczby 9 na trzy różne składniki. Ciemne kropki odpowiadają rozkładowi liczby 6 na 3 mogące się powtarzać składniki

Tw. 4.9. *Liczby $q(n, k)$ spełniają rekurencję*

$$q(n, k) = q(n - k, k - 1) + q(n - k, k). \quad (4.24)$$

Dowód: Twierdzenie wynika z rekurencji (4.20) i związku (4.23). \square

Trójkąt dla liczb $q(n, k)$ skonstruowany zgodnie z regułą (4.24) przedstawiony jest na rys. 4.13.

Na koniec podrozdziału przedstawimy jeszcze jeden ciekawy problem, ukażający po raz kolejny, jak potężnym narzędziem w analizie kombinatorycznej są funkcje tworzące. Udowodnimy z ich pomocą pewne intuicyjnie nieoczywiste twierdzenie:

Tw. 4.10. *Daną liczbę naturalną możemy na tyle samo sposobów rozłożyć na składniki nieparzyste co na składniki różne.*

Zacznijmy od przykładu, pamiętamy, że kolejność występowania składnika nie jest istotna:

					1							
					1	0						
					1	1	0					
					1	1	0	0				
					1	2	0	0	0			
					1	2	1	0	0	0		
					1	3	1	0	0	0	0	
					1	3	2	0	0	0	0	
					1	4	3	0	0	0	0	
					1	4	4	1	0	0	0	

Rysunek 4.13: Trójkąt liczby partycji n na k różnych składników, $q(n, k)$, gdzie $n = 1, 2, \dots$ numeruje wiersze, a $k = 1, 2, \dots, n$ kolumny. Rekurencja 4.24 prowadzi do następującej konstrukcji: dana liczba jest sumą liczb z wiersza o k pozycji wyżej, w kolumnach k i $k - 1$. Na przykład $q(10, 3) =$ jest sumą $q(7, 2) + q(7, 3) = 3 + 1$. Suma wyrazów w każdym wierszu daje całkowitą liczbę podziałów liczby n na składniki różne, $q(n)$.

składniki nieparzyste	składniki różne
$7=1+1+1+1+1+1+1$	$7=1+2+4$
$7=3+1+1+1+1$	$7=1+6$
$7=3+3+1$	$7=2+5$
$7=5+1+1$	$7=3+4$
$7=7$	$7=7.$

W każdym przypadku mamy istotnie po tyle samo (pięć) możliwości.

Dowód: Funkcja tworząca dla rozkładu na mogące się powtarzać składniki nieparzyste ma postać

$$F_1(z) = \frac{1}{1-z} \frac{1}{1-z^3} \frac{1}{1-z^5} \frac{1}{1-z^7} \dots$$

natomast funkcja tworząca dla rozkładu na różne składniki naturalne to

$$F_2(z) = (1+z)(1+z^2)(1+z^3)(1+z^4)(1+z^5)(1+z^6)(1+z^7) \dots$$

Musimy pokazać, że zachodzi równość

$$F_1(z) = F_2(z). \quad (4.25)$$

Skorzystamy z oczywistego faktu $(1-z^k)(1+z^k) = 1-z^{2k}$. Przemnożmy obie strony (4.25) przez $(1-z)$. Otrzymujemy

$$\begin{aligned} & \frac{1}{1-z^3} \frac{1}{1-z^5} \frac{1}{1-z^7} \dots = (1-z^2)(1+z^2)(1+z^3)(1+z^4)(1+z^5)(1+z^6) \dots \\ & = (1-z^4)(1+z^3)(1+z^4)(1+z^5)(1+z^6) \dots \\ & = (1+z^3)(1-z^8)(1+z^5)(1+z^6) \dots = (1+z^3)(1+z^5)(1+z^6) \dots \end{aligned}$$

Po prawej stronie pojawił się najpierw czynnik $(1-z^2)$, który razem z $(1+z^2)$ dał $(1-z^4)$, ten z kolei po wymnożeniu przez $(1+z^4)$ dał $(1-z^8)$ itd. W efekcie po prawej stronie zniknęły wszystkie czynniki postaci $(1+z^n)$, gdzie $n = 2, 4, 8, 16, \dots = 2^k$, $k = 1, 2, 3, \dots$. Następnie powtarzamy procedurę, wymnażając obie strony przez $(1-z^3)$. W wyniku znikają czynniki postaci $(1+z^n)$ dla $n = 3 \cdot 2^k$. Potem wymnażamy przez $(1-z^5)$ itd. W rezultacie po prawej stronie pozbywamy się czynników o potęgach z równych $m \cdot 2^k$, gdzie m jest kolejną liczbą nieparzystą. Czynniki te dla kolejnych wartości m są wypisane jawnie w tabeli 4.1. Istota naszego dowodu polega na tym, że tabela zawiera *wszystkie* liczby naturalne, w dodatku każdą dokładnie jeden raz! Wynika to z podstawowego twierdzenia algebra, że każdą liczbę naturalną można jednoznacznie rozłożyć na czynniki pierwsze. Wyciągając wszystkie potęgi dwójki, 2^k , otrzymujemy liczbę nieparzystą m , co właśnie daje postać $m \cdot 2^k$. Pokazaliśmy zatem równość (4.25).

□

Tabela 4.1: Czynniki z dowodu Tw. 4.10

m	$m \cdot 2^k$
1	1, 2, 4, 8, 16, 32, ...
3	3, 6, 12, 24, 48, ...
5	5, 10, 20, 40, ...
7	7, 14, 28, ...
9	9, 18, 36, ...
11	11, 22, ...

4.6 Kompendium najczęstszych problemów kombinatorycznych

Rozważania tego rozdziału oraz rozdz. 2 podsumowujemy zestawieniem wzorów dla najczęściej występujących zagadnień kombinatorycznych. Pierwsza grupa to problem wyboru k obiektów spośród n rozróżnialnych obiektów z kolejnością istotną lub nie. Dodatkowo dopuszcamy lub nie powtórzenia wyboru tego samego elementu. Mamy więc łącznie cztery możliwości przedstawione w tabeli 4.2. Druga grupa problemów to wkładanie n obiektów (piłek), rozróżnialnych lub nie, do k wiaderek, rozróżnialnych lub nie. Dodatkowo dopuszcamy lub nie możliwość pozostawienia niektórych wiaderek pustych. Osiem możliwości przedstawionych jest w tabeli 4.3. Symbol $\{.\}$ oznacza tu liczby Stirlinga drugiego rodzaju z podrozdziału 4.2, a p_k^n liczbę k -elementowych partycji liczby n z podrozdziału 4.5.

Wzory w czwartym rzędzie tabeli 4.2 oraz piątym rzędzie tabeli 4.3 są tematem zadań 2.8 i 2.7, natomiast wzory z drugiego i szóstego rzędu tabeli 4.3 zadań 4.4 i 4.5. Pozostałe wzory są elementarne.

Tabela 4.2: Wybieranie k obiektów z n rozróżnialnych obiektów

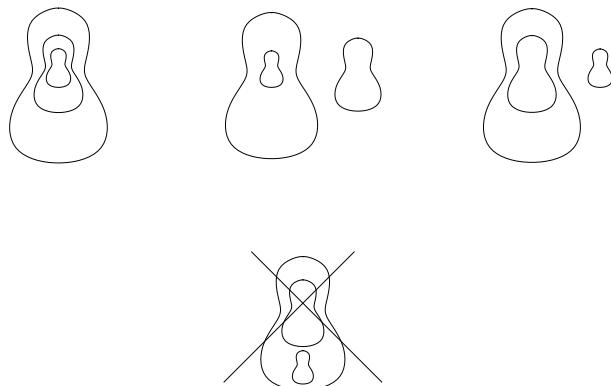
Kolejność istotna	Możliwość powtórzeń	Nazwa	Liczba sposobów
tak	tak	wariacja z powtórzeniami	n^k
tak	nie	wariacja bez powtórzeń $n = k$ – permutacja	$\frac{n!}{(n-k)!}$ $n!$
nie	nie	kombinacja	$\binom{n}{k}$
nie	tak	kombinacja z powtórzeniami	$\binom{n+k-1}{k}$

Tabela 4.3: Wkładanie n piłek do k wiaderek

Piłki rozróżnialne	Wiaderka rozróżnialne	Wiaderka mogą być puste	Liczba sposobów
tak	tak	tak	n^k
tak	tak	nie	$k! \left\{ \begin{array}{c} n \\ k \end{array} \right\}$
tak	nie	tak	$\left\{ \begin{array}{c} n \\ 1 \end{array} \right\} + \cdots + \left\{ \begin{array}{c} n \\ k \end{array} \right\}$
tak	nie	nie	$\left\{ \begin{array}{c} n \\ k \end{array} \right\}$
nie	tak	tak	$\left(\begin{array}{c} n+k-1 \\ n \end{array} \right)$
nie	tak	nie	$\left(\begin{array}{c} n-1 \\ k-1 \end{array} \right)$
nie	nie	tak	$p_1^n + \cdots + p_k^n$
nie	nie	nie	p_k^n

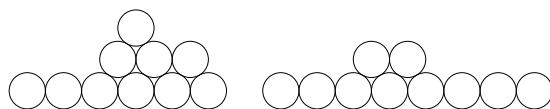
Ćwiczenia

- 4.1. Udowodnij wzory (4.2-4.5), korzystając z kombinatorycznej definicji liczb Stirlinga.
- 4.2. Udowodnij, że $\sum_{k=0}^n \left[\begin{array}{c} n \\ k \end{array} \right] = n!$.
- 4.3. Matrioszka to zabawka, składająca się z n coraz mniejszych, wchodzących jedna w drugą lalek. (a) Na ile różnych sposobów można złożyć te lalki? Legalne konfiguracje to takie, że po otwarciu każdej lalki w jej wnętrzu widać co najwyżej jedną lalkę (która może w swoim środku zawierać inne lalki), zob. rys. 4.14. (b) Patrząc na matrioszki, nie widzimy, co jest we wnętrzu danej lalki. Na ile różnych tak samo „na zewnątrz” wyglądających sposobów można złożyć lalki?
- 4.4. Na ile sposobów można rozmieścić n rozróżnialnych piłek w k rozróżnialnych wiaderkach tak, by żadne wiaderko nie było puste?
- 4.5. Na ile sposobów można rozmieścić n nierożróżnialnych piłek w k rozróżnialnych wiaderkach tak, by żadne wiaderko nie było puste?
- 4.6. Rozważ ścieżki na kwadracie $n \times n$ prowadzące od dolnego lewego do prawnego górnego rogu. Ograniczenia są następujące: 1) ścieżki nie mogą opadać, tzn. w każdym kolejnym ruchu możemy posuwać się w górę lub pozostać na tej samej wysokości, oraz 2) ścieżki nie przecinają przekątnej idącej od dolnego lewego do prawnego górnego rogu. Ile jest takich ścieżek?



Rysunek 4.14: Przykłady legalnych (górnny rząd) i nielegalnych (dolny rząd) konfiguracji matroszek z zad. 4.3

- 4.7. Pokaż, że liczba rozkładów n na nie więcej niż k składników jest równa liczbie rozkładów liczby $n+k$ na dokładnie k składników.
- 4.8. Udowodnij, że liczba partycji, w której nie powtarzają się parzyste składniki, jest równa liczbie partycji, w której żadna liczba nie powtarza się więcej niż 3 razy.
- 4.9. Wykaż, że liczba rozkładów liczby n niezawierających składnika 1 jest równa $p_n - p_{n-1}$.
- 4.10. Udowodnij równania (4.21).
- 4.11. Wyprowadź funkcję tworzącą dla liczb Stirlinga pierwszego rodzaju.
- 4.12. Na ile sposobów kioskarz może rozmienić n złotych, posługując się monetami o nominałach 1, 2 i 5 zł?
- 4.13. Na ile sposobów może kioskarz wydać 66 zł, mając do dyspozycji po 9 monet 5, 2 i 1 zł?
- 4.14. Rozważ zabawę, polegającą na układaniu jednogroszówek z następującymi zasadami: rzędy układane są poziomo jeden nad drugim (dopuszczalne jest ułożenie tylko w jednym rzędzie), monety w danym rzędzie muszą do siebie przylegać, a każda moneta w rzędzie wyższym musi przylegać do dwóch monet rzędu niższego (zob. rys. 4.15). (a) Na ile sposobów można ułożyć jednogroszówki, jeśli w dolnym rzędzie jest ich n ? (b) Na ile sposobów można ułożyć łączną liczbę n jednogroszówek? (problem pochodzi z [44])?



Rysunek 4.15: Przykładowe legalne ułożenia dziesięciu jednogroszówek z ćw. 4.14

Rozdział 5

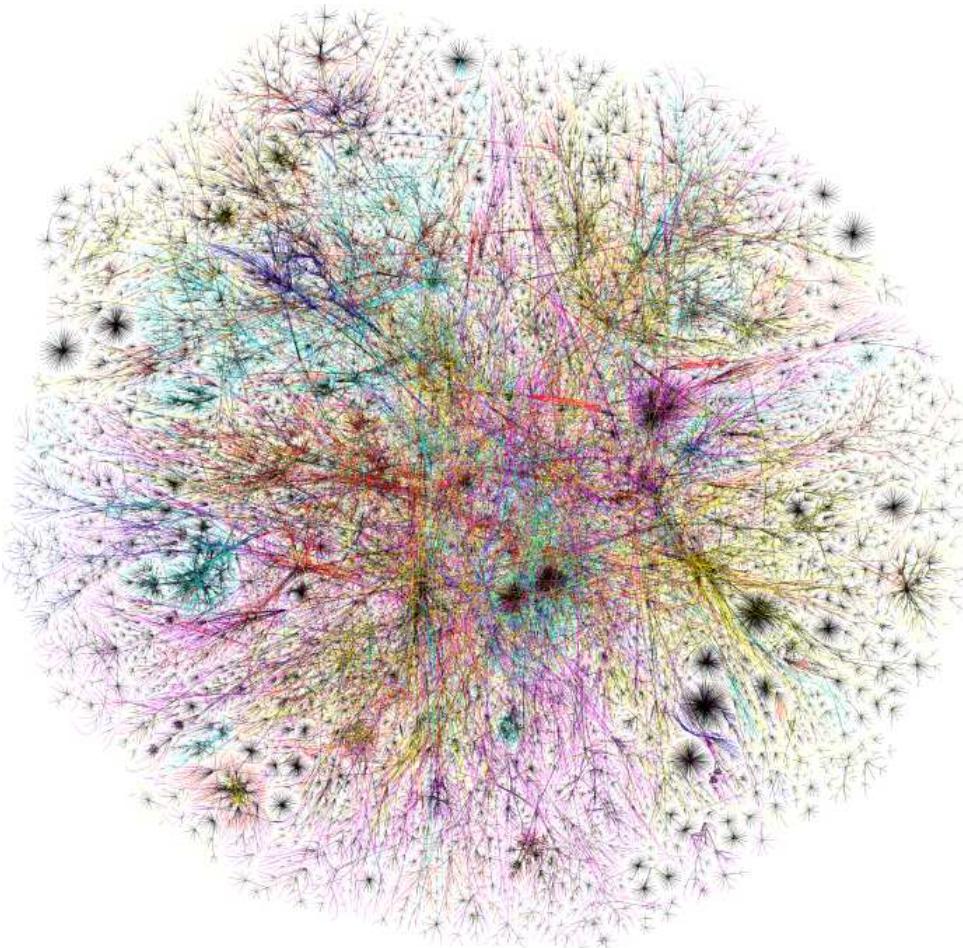
Grafy

W tym kluczowym dla wykładu rozdziale przedstawimy najważniejsze elementy *teorii grafów i sieci*. Omówimy podstawowe typy grafów, jak grafy eulerowskie, hamiltonowskie, planarne tudzież związane z nimi twierdzenia. Na podstawie wzoru wielościanu Eulera zrozumiemy konstrukcję brył platońskich. Przedstawiemy popularne algorytmy przeszukiwania grafów oraz znajdowania minimalnego drzewa spinającego i najkrótszej ścieżki w grafach z wagami. Opiszemy *drzewo Steinera* jako istotne dla nafciarza z Teksasu, a także problem *Małego Świata* czy *sześciu stopni oddalenia*, występujący w sieciach społecznych. Przyjrzymy się też zagadnieniom związanym z kolorowaniem grafów, jak problem układania harmonogramu zajęć na uczelni oraz słynne twierdzenie o kolorowaniu map co najwyżej czterema kolorami. Ukażemy drzewa binarne jako struktury użyteczne dla przechowywania informacji i zapisu wyrażeń arytmetycznych, co dzięki notacji polskiej pozwoliło firmie Hewlett-Packard stworzyć kalkulator bez nawiasów i znaku równości na klawiaturze! Przy okazji pieczenia szarlotki przedstawimy *sieci zdarzeń*, omówimy też *przepływy w sieciach* i związane z nimi twierdzenia i algorytmy.

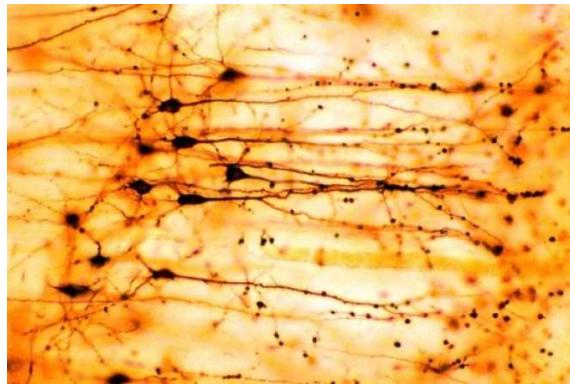
Aby docenić ważkość tych zagadnień dla wielu dziedzin nauki, od informatyki i telekomunikacji po neurobiologię i socjologię, spójrzmy najpierw na rys. 5.1-5.3. Rysunek 5.1 przedstawia sieć połączeń internetowych między routerami ok.

2003 r. Routery (punkty czy też *wierzchołki* na rysunku) połączone są między sobą liniami (krawędziami), co obrazuje ich fizyczne połączenie linią przesyłową. Podobnie skomplikowana struktura realizowana jest w sieciach neuronowych w mózgu (zob. rys. 5.2, ukazujący wycinek mózgu małpy pobudzony światłem lasera). Potężną strukturę, o ponad 800 mln użytkowników – wierzchołków, stanowi sieć użytkowników portalu Facebook. Rysunek 5.3 obrazuje strukturę znajomości w obrębie tej społeczności. Linie łączą miasta, z których osoby deklarują się jako przyjaciele, natomiast grubość linii jest proporcjonalna do liczby tych przyjaciół.

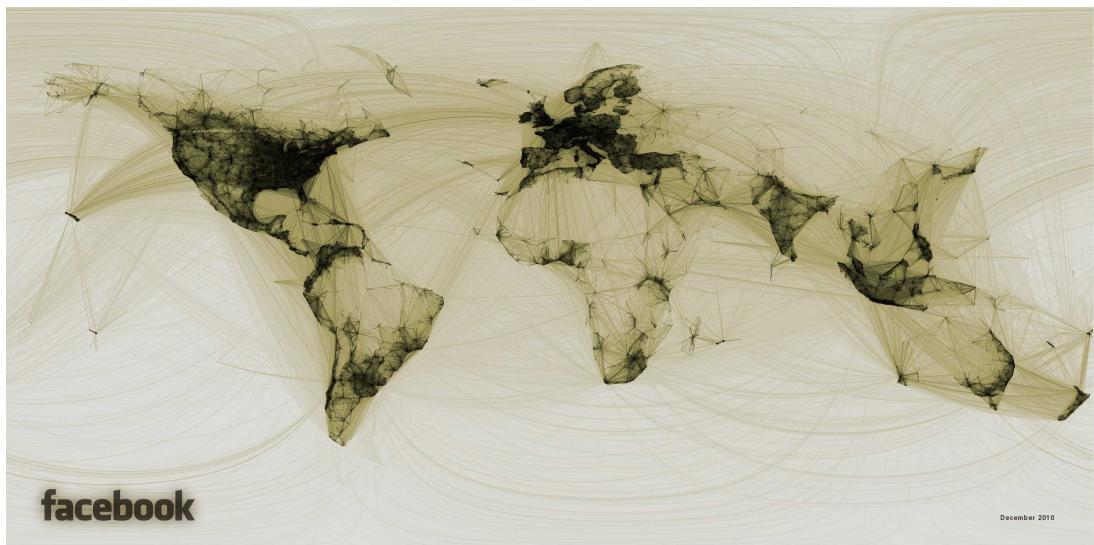
Ramy dla analizy wspomnianych powyżej bardzo złożonych układów i zchodzących w nich procesów tworzy właśnie teoria grafów i sieci, której podstawy omawiamy w niniejszym rozdziale.



Rysunek 5.1: Graf przedstawiający połączenia internetowe między routery ok. 2003 r.
(<http://visualgadgets.blogspot.com/2008/06/graphs-and-networks.html>)



Rysunek 5.2: Graf połączeń neuronów w wycinku mózgu małpy
(<http://www.wired.com/wiredscience/2009/04/lasercontrolledhumans>)



Rysunek 5.3: Graf wizualizujący znajomości w portalu Facebook. Linie łączą miasta, z których osoby deklarują się jako przyjaciele. Grubość linii jest proporcjonalna do liczby przyjaciół
(<http://www.facebook.com/notes/facebook-engineering/visualizing-friendships>)

5.1 Spacer Eulera

Rozpoczniemy, jak wiele innych podręczników teorii grafów, od anegdotycznego spaceru Eulera w 1736 r. po Królewcu. Plan twierdzi z tego okresu jest przedstawiony na rys. 5.4. Problem mostów królewieckich brzmi: *Czy można odbyć spacer po mieście w taki sposób, aby przez każdy z siedmiu mostów przejść dokładnie raz i wrócić do punktu wyjścia?* Euler rozwiązał ten problem i uogólnił go, stawiając podwaliny pod teorię grafów.



Rysunek 5.4: Królewiec i jego siedem mostów w XVIII w.
(<http://www.amt.edu.au/koenigs.html>)

Pierwszym krokiem jest przerysowanie sytuacji z rys. 5.4 w schematyczny sposób, bez zbędnych szczegółów, tak jak na rys. 5.5. Obszarom z mapy oddzielonym Pregołą i jej rozwidleniami odpowiadają punkty oznaczone literami – są to wierzchołki grafu. Mostom, które łączą te obszary, odpowiadają linie łączące

wierzchołki – są to krawędzie grafu. Zanim wcielimy się w Eulera, podajmy kilka precyzyjnych definicji.

Def. 5.1. Grafem nieskierowanym (lub po prostu grafem) G nazywamy trójkę składającą się z 1) skońzonego niepustego zbioru wierzchołków, oznaczonego jako W , 2) skońzonego zbioru krawędzi K oraz 3) funkcji $\gamma : K \rightarrow P(W)$, gdzie $P(W)$ jest zbiorem jedno- i dwuelementowych podzbiorów W .

Co to oznacza?

Przykład 5.1. W problemie mostów zbiór wierzchołków to $W = \{A, B, C, D\}$ (obszary miasta), zbiór krawędzi to $K = \{1, 2, 3, 4, 5, 6, 7\}$ (mosty), a funkcja γ , dana jako

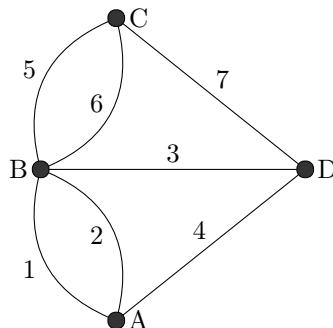
$$\begin{aligned}\gamma(1) &= \{A, B\}, & \gamma(2) &= \{A, B\}, \\ \gamma(3) &= \{B, D\}, & \gamma(4) &= \{A, D\}, \\ \gamma(5) &= \{B, C\}, & \gamma(6) &= \{B, C\}, \\ \gamma(7) &= \{C, D\},\end{aligned}$$

określa, które dwa obszary (litery) łączy dany most (liczby 1–7).

Zbiory wierzchołków i krawędzi danego grafu G będziemy często oznaczać jako $W(G)$ oraz $K(G)$.

Def. 5.2. Dwa wierzchołki są sąsiednie, jeśli istnieje łącząca je krawędź. Dwie krawędzie są sąsiednie, jeśli mają przynajmniej jeden wspólny wierzchołek.

Na rys. 5.5 wierzchołki A i B są sąsiednie, a A i C nie są sąsiednie, krawędzie 1 i 4 są sąsiednie, a 1 i 7 nie są sąsiednie.



Rysunek 5.5: Problem mostów królewieckich. Krawędzie ponumerowano tak samo, jak mosty na rycinie 5.4. Wierzchołki oznaczone (inaczej niż na rycinie) literami odpowiadają częściom miasta oddzielonym rzeką i jej rozwidleniami

W naszym przykładzie funkcja γ odwzorowuje w zbiór dwuelementowych podzbiorów. W ogólności możemy mieć sytuację taką, jak na rys. 5.6(a), gdzie mamy pętle wokół pojedynczego wierzchołka. Wtedy przeciwdziedzina funkcji γ zawiera jednoelementowe podzbiory W , np. dla rys. 5.6(a) mamy $\gamma(a) = \{A\}$.

Funkcję γ można przedstawić w postaci *macierzy incydencji*.

Def. 5.3. *Macierz incydencji* M grafu G to macierz, której wiersze odpowiadają krawędziom, a kolumny wierzchołkom G . Jeśli krawędź i ma początek lub koniec w wierzchołku j , to $M_{ij} = 1$, w przeciwnym przypadku $M_{ij} = 0$.

Dla problemu mostów królewieckich mamy

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad (5.1)$$

gdzie kolumny odpowiadają wierzchołkom (obszarom miasta) A, B, C i D , a wiersze mostom 1–7. Na przykład wyczytujemy, że most 5 łączy wierzchołki B i C . Jeśli graf zawiera pętle wokół pojedynczego wierzchołka i , wtedy element $M_{ii} = 2$.

Innym, bardziej zwartym zapisem tej samej informacji o grafie jest *macierz sąsiedztwa*.

Def. 5.4. *Macierz sąsiedztwa* S grafu G to macierz, w której zarówno wiersze, jak i kolumny odpowiadają wierzchołkom, a element $S_{ij} = n$, jeśli wierzchołki i oraz j są połączone n krawędziami, natomiast $S_{ij} = 0$, gdy nie są połączone.

Macierz sąsiedztwa dla problemu mostów królewieckich ma postać

$$S = \begin{pmatrix} 0 & 2 & 0 & 1 \\ 2 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \quad (5.2)$$

W szczególności widzimy, że wierzchołki A i B są połączone dwiema krawędziami. Jeśli graf zawiera pętlę wokół pojedynczego wierzchołka i , wtedy element $S_{ii} = 1$.

Def. 5.5. *Stopień wierzchołka* grafu to całkowita liczba doczepionych doń krawędzi. Krawędź zapętlająca się wokół tego samego wierzchołka liczymy podwójnie.



Rysunek 5.6: Przykłady grafu nieskierowanego (a) i skierowanego (b)

Dla grafu mostów królewieckich wierzchołek B ma stopień 5 (z obszaru B wychodzi 5 mostów), pozostałe wierzchołki mają stopień 3.

Lemat 5.1 (o uściskach dloni). *Suma stopni wszystkich wierzchołków grafu jest liczbą parzystą równą podwojonej liczbie krawędzi.*

Dowód: Dowód odzwierciedla nazwę tego lematu. W grupie osób (wierzchołki) doszło do j uścisków dloni (krawędzie). Ponieważ w każdym uścisku biorą udział dwie dłonie, to łącznie wszystkie dlonie uścisnęły się $2j$ razy. \square

Spacer Eulera po mostach królewieckich mógł się odbywać w dowolnym kierunku, co jest oczywiste – piesi zazwyczaj nie podlegają ruchowi jednokierunkowemu! W wielu sytuacjach mamy jednak w naturalny sposób do czynienia z kierunkowością, czego przykładem są drogi jednokierunkowe, diagramy ukazujące ewolucję w czasie, np. jajo \rightarrow pisklę \rightarrow kurczak itd. W związku z tym wprowadzamy pojęcie grafu skierowanego:

Def. 5.6. *Grafem skierowanym (zorientowanym lub digrafem) G nazywamy trójkę składającą się z 1) skończonego niepustego zbioru W wierzchołków, 2) skończonego zbioru krawędzi K oraz 3) funkcji $\gamma : K \rightarrow W \times W$.*

W tym przypadku γ odwzorowuje w iloczyn kartezjański zbioru wierzchołków, tzn. w zbiór wszystkich par utworzonych z elementów W . Innymi słowy, istotna jest kolejność. Krawędzie grafu skierowanego przedstawiamy graficznie w postaci strzałek, tak jak na rys. 5.6(b), gdzie np. $\gamma(a) = (A, D)$. Dwa wierzchołki mogą być połączone kilkoma krawędziami. Zauważmy, że nie musimy rysować strzałek dla pętli wokół pojedynczego wierzchołka. I jeszcze jedna uwaga: graf skierowany nie oznacza, że wszystkie możliwe przejścia między wierzchołkami są jednokierunkowe. Wystarczy połączyć wierzchołki krawędziami skierowanymi idącymi w przeciwnie strony (dwa pasy ruchu na jezdni), aby możliwe było przejście w obie strony. Przykładem są wierzchołki C i D na rys. 5.6(b), połączone strzałkami idącymi w obie strony.

Macierz sąsiedztwa S dla grafu skierowanego o n wierzchołkach to macierz $n \times n$, gdzie S_{ij} jest liczbą krawędzi idących od wierzchołka i do wierzchołka j . Dla grafu z rys. 5.6(b) mamy więc

$$S = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (5.3)$$

W oczywisty sposób macierz S niesie pełną informację o grafie. Widzimy, że graf ma 4 wierzchołki (liczba wierszy i kolumn), 6 krawędzi (suma wszystkich wyrazów macierzy) oraz jedną pętlę (jeden wyraz na przekątnej). W odróżnieniu od macierzy sąsiedztwa dla grafu nieskierowanego, macierz S dla grafu skierowanego w ogólności *nie jest symetryczna*.

Zauważmy, że zarówno macierz incydencji, jak i macierz sąsiedztwa zawierają wiele zer i w ten sposób „marnują miejsce”. Macierz incydencji dla grafu o k krawędziach i n wierzchołkach zawiera łącznie kn liczb, a macierz sąsiedztwa n^2 liczb. Bardziej wydajnym sposobem przechowywania pełnej informacji o grafie są tzw. *listy incydencji*. Listy (liczba mnoga) są sporządzane w następujący sposób: bierzemy kolejno wierzchołki grafu i dla każdego wypisujemy sąsiadujące z nim wierzchołki. Na przykład dla grafu z rys. 5.6(b) listy incydencji są następujące:

$$L = \left(\begin{array}{c|cc} A & B & D \\ B & B & \\ C & D & D \\ D & C & \end{array} \right), \quad (5.4)$$

co oznacza, że z A możemy pójść, zgodnie z kierunkiem strzałek, do B i D , z B możemy pójść po pętli do B , z C mamy dwie krawędzie do D , a z D możemy udać się do C . Dla grafu nieskierowanego z rys. 5.6(a) mamy

$$L = \left(\begin{array}{c|ccc} A & A & B & \\ B & A & B & C \\ C & B & D & \\ D & C & & \end{array} \right). \quad (5.5)$$

Zauważmy, że w listach incydencji możemy opuścić lewą kolumnę, jeśli umówimy się, że ustawiamy wierzchołki w porządku leksykograficznym. Na końcu każdego wiersza musimy natomiast umieścić znacznik końca linii, ponieważ wiersze mają różną długość. Można się łatwo przekonać, że stosując zapis list incydencji dla grafu skierowanego potrzebujemy $k + n$ liczb, a dla grafu nieskierowanego $2k + n$ liczb.

A teraz seria definicji:

Def. 5.7. Krawędź o początku i końcu w tym samym wierzchołku nazywamy pętlą.



Def. 5.8. Krawędzie o wspólnym początku i wspólnym końcu nazywamy krawędziami wielokrotnymi.



W licznych zastosowaniach występują grafy bez pętli i krawędzi wielokrotnych.

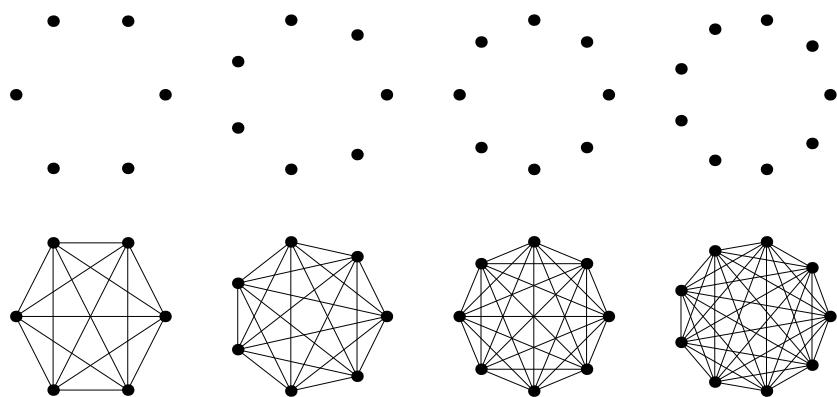
Def. 5.9. Graf bez pętli i krawędzi wielokrotnych nazywamy grafem prostym.

Def. 5.10. Graf pusty to graf bez krawędzi, czyli składający się z wierzchołków izolowanych. Graf pusty o n wierzchołkach oznaczamy jako N_n (rys. 5.7).

Def. 5.11. Graf pełny to graf prosty, w którym każde dwa wierzchołki są połączone krawędzią. Graf pełny o n wierzchołkach oznaczamy jako K_n (rys. 5.7).

Są to zatem grafy „nikt z nikim” i „każdy z każdym”. Jest oczywiste, że w grafie K_n stopień każdego wierzchołka wynosi $n - 1$ oraz graf posiada $n(n - 1)/2$ krawędzi (liczba par wierzchołków). Jest to największa liczba krawędzi, jaką może posiadać graf prosty o n wierzchołkach. Rysunek 5.7 pokazuje kilka grafów pełnych dla kolejnych wartości n .

Def. 5.12. Ciąg składający się z wierzchołka początkowego, krawędzi sąsiedniej do tego wierzchołka oraz kolejnych sąsiednich krawędzi nazywamy drogą (ściezką, marszrutą). Długością drogi jest liczba składających się na nią krawędzi.



Rysunek 5.7: Grafy puste $N_6 - N_9$ i pełne $K_6 - K_9$

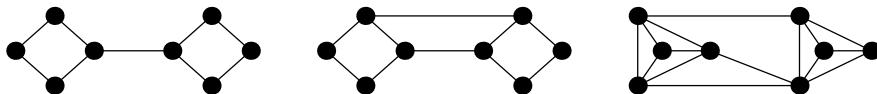
Nietrudno się przekonać, że ta definicja określa drogę w sposób jednoznaczny dla dowolnego grafu. W grafie prostym możemy też wyspecyfikować drogę poprzez podanie ciągu sąsiednich wierzchołków. Jest tak, ponieważ przy braku krawędzi wielokrotnych jest tylko jedna możliwość wyboru krawędzi pomiędzy dwoma sąsiednimi wierzchołkami.

Def. 5.13. *Graf jest spójny, jeśli każde dwa różne wierzchołki są połączone jakąś drogą.*

Mówiąc potocznie, graf spójny jest „w jednym kawałku”.

Def. 5.14. *Graf jest k -spójny, jeśli graf powstały przez usunięcie dowolnych $k - 1$ krawędzi pozostaje spójny, a istnieje takich k krawędzi, że ich usunięcie generuje graf niespójny.*

Przykłady pokazane są poniżej. Usunięcie środkowej krawędzi z grafu po lewej stronie powoduje jego „rozspójnienie”, więc graf jest jednospójny. Podobnie, środkowy graf jest dwuspójny, a graf po prawej 3-spójny.



Tw. 5.1. *Liczba krawędzi k w grafie spójnym prostym o n wierzchołkach spełnia nierówności*

$$n - 1 \leq k \leq \frac{n(n - 1)}{2}. \quad (5.6)$$

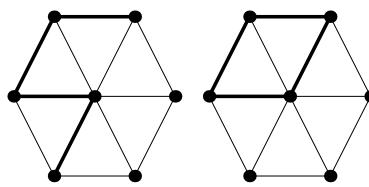
Dowód: Górnne ograniczenie wynika z faktu, że $n(n - 1)/2$ jest liczbą krawędzi grafu pełnego. Dolne ograniczenie dowodzimy indukcyjnie. Twierdzenie zachodzi dla grafu pustego o jednym wierzchołku. Dodanie każdego kolejnego wierzchołka musi wiązać się z dodaniem co najmniej jednej krawędzi. Zachodzą tu dwa przypadki. Albo dodajemy wierzchołek na istniejącej krawędzi, wtedy wierzchołek dzieli tę krawędź na dwie krawędzie, wobec czego n i k wzrastają o 1 i teza pozostaje spełniona. W drugim przypadku dodajemy wierzchołek poza istniejącą krawędzią. Ponieważ graf musi być spójny, musimy też dodać krawędź mającą koniec w istniejącym wcześniej wierzchołku. Ponownie n i k wzrastają o 1 i teza jest spełniona, co kończy dowód. \square

Z Tw. 5.1 wynika, że $k + n \leq n(n + 1)/2 \leq n^2$ (poprzez dodanie n po obu stronach prawej nierówności (5.6)) oraz $2k + n \leq n^2$ (poprzez przemnożenie obu

stron nierówności (5.6) przez 2 i dodanie n), zatem listy incydencji zawierają nie więcej liczb niż macierze sąsiedztwa.

Def. 5.15. Drogą prostą nazywamy drogę, która ma wszystkie krawędzie różne. Drogą zamkniętą nazywamy drogę, która ma ten sam początek i koniec (w przeciwnym razie droga jest otwarta). Zamkniętą drogę prostą nazywamy cyklem.

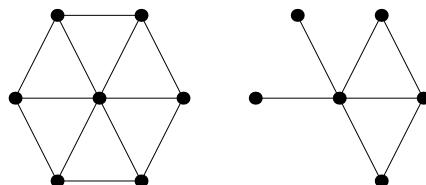
Poniższy rysunek pokazuje (poprzez wytłuszczenie krawędzi) przykład drogi prostej (lewa strona) i cyklu (prawa strona) w grafie.



Def. 5.16. Odległość dwóch wierzchołków w grafie to długość najkrótszej drogi, która je łączy. Średnica grafu to największa odległość dwóch wierzchołków tego grafu. Obwód grafu to długość najkrótszego cyklu.

Średnica grafów narysowanych powyżej wynosi 2, a obwód 3.

Def. 5.17. Podgrafem grafu G nazywamy graf, którego zbiór wierzchołków zawiera się w $W(G)$, a zbiór krawędzi w $K(G)$.



Mówiąc bardziej obrazowo, jak ukazano na powyższym rysunku, podgraf powstaje z danego grafu poprzez usunięcie pewnej liczby wierzchołków (oczywiście musimy też jednocześnie usunąć krawędzie wychodzące z usuniętych wierzchołków) oraz pewnej liczby krawędzi.

Def. 5.18. Podgraf grafu G , który jest grafem pełnym, nazywamy kliką.

Nazwa pochodzi oczywiście od analogii do sytuacji społecznej, gdzie w pewnej grupie osób istnieje podgrupa, w której każdy zna (rozmania, utrzymuje kontakty, ma konszachty, popiera politycznie, ...) każdego. W grafie po lewej stronie rys. 5.8 mamy klikę o czterech wierzchołkach, a w grafie po prawej stronie trzy kliki o trzech wierzchołkach. Stosowana nomenklatura sygnalizuje stosowność teorii grafów w naukach społecznych.

Teraz jesteśmy już uzbrojeni w stosowny aparat pojęciowy, aby powrócić do przechadzki Eulera po Królewcu.

Def. 5.19. *Drogę prostą zawierającą wszystkie krawędzie grafu nazywamy drogą Eulera. Cykl zawierający wszystkie krawędzie grafu nazywamy cyklem Eulera. Graf posiadający cykl Eulera nazywamy eulerowskim. Graf posiadający drogę Eulera, ale nie posiadający cyklu Eulera, nazywamy półeulerowskim.*

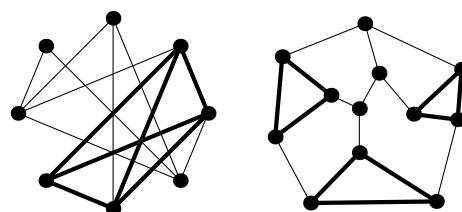
Tak więc problem mostów królewieckich można sformułować następująco: *Czy graf z rys. 5.5 zawiera cykl Eulera?* Odpowiedź okazuje się bardzo prosta:

Tw. 5.2 (Eulera). *Graf spójny jest eulerowski wtedy i tylko wtedy, gdy wszystkie jego wierzchołki są stopnia parzystego.*

A zatem **problem mostów królewieckich nie ma rozwiązania**, ponieważ wierzchołki grafu z rys. 5.5 są stopnia nieparzystego!

Dowód: Twierdzenie, że jeśli graf ma cykl Eulera, to jego wierzchołki są stopnia parzystego, jest oczywiste, jeśli popatrzymy na problem w następujący sposób: narysujmy punkty (wierzchołki) na kartce papieru. Na początku stopień każdego wierzchołka jest zerowy, więc parzysty. Weźmy teraz ołówek i narysujmy *bez odrywania* ołówka od papieru linię łączącą nasze punkty zgodnie z cyklem Eulera. Linia wychodzi z punktu A , więc w tym momencie stopień wierzchołka A wzrasta o jeden i staje się nieparzysty. Następnie odwiedzamy kolejne wierzchołki, a każde odwiedziny zwiększają stopień odwiedzonego wierzchołka o dwa, ponieważ jedna krawędź wchodzi i za chwilę wychodzi nowa. Kończymy tam gdzie zaczeliśmy, konstruujemy bowiem cykl Eulera, a więc w A . Wobec tego zwiększamy stopień A o 1 i w ten sposób stopień A staje się parzysty. Widzimy więc, że wszystkie wierzchołki tak skonstruowanego grafu są parzyste. W ten sposób pokazaliśmy, że każdy graf eulerowski ma wszystkie wierzchołki stopnia parzystego.

Pozostaje pokazać zachodzenie twierdzenia w drugą stronę, tj. że graf spójny G o wszystkich wierzchołkach stopnia parzystego jest eulerowski. Najpierw udowodnimy użyteczny lemat.



Rysunek 5.8: Przykłady grafów z zaznaczonymi (poprzez wytłuszczenie krawędzi) klikami

Lemat 5.2. *Każdy graf o wierzchołkach stopnia co najmniej 2 zawiera cykl.*

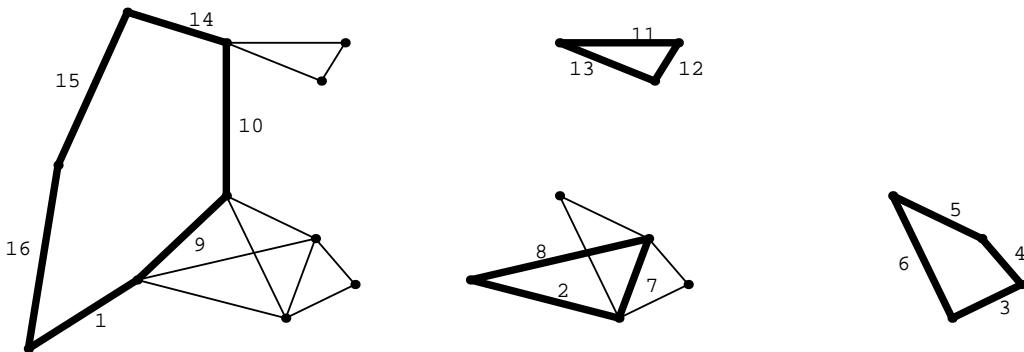
Dowód: Jeżeli graf zawiera pętle lub krawędzie wielokrotne, lemat jest oczywisty. Jeżeli graf jest grafem prostym, to postępujemy w następujący sposób: startujemy z wierzchołka w_1 , idziemy do wierzchołka w_2 różnego od w_1 , z wierzchołka w_i , $i \geq 2$, indukcyjnie idziemy do wierzchołka w_{i+1} różnego od w_i i w_{i-1} . Fakt, że wybór taki jest możliwy, wynika stąd, że stopień każdego wierzchołka jest co najmniej 2 oraz z założenia, że graf nie ma pętli (nie wracamy do w_i) ani krawędzi wielokrotnych (nie wracamy do w_{i-1}). Ponieważ liczba wierzchołków grafu n jest skończona, to po co najwyżej n krokach wierzchołek w_1 się powtórzy. W ten sposób uzyskujemy pożądany cykl. \square

Teraz wracamy do dowodu Tw. Eulera. Graf G jest spójny i o wierzchołkach stopnia parzystego, więc stopień każdego wierzchołka wynosi co najmniej 2 i na mocy lematu graf ma pewien cykl C . Jeśli C zawiera każdą krawędź grafu, dowód jest zakończony. W przeciwnym razie usuwamy krawędzie cyklu C oraz powstałe wierzchołki izolowane z G . W efekcie powstaje graf G' (w ogólności niespójny), którego każdy wierzchołek jest parzysty. Parzystość wynika z faktu, że usuwając krawędzie cyklu C z wierzchołków, G odejmowaliśmy po dwie krawędzie. Dla każdej ze spójnych składowych G' powtarzamy indukcyjnie opisaną konstrukcję. Ponieważ graf jest skończony, konstrukcja też musi się zakończyć. W rezultacie mamy graf G rozłożony na krawędziowo rozłączne cykle, tj. cykle nieposiadające wspólnych krawędzi. Zauważmy, że w oczywisty sposób każdy punkt G należy do jakiegoś cyklu. Możemy wreszcie podać konstrukcję cyklu Eulera dla grafu G , która jest następująca: startujemy z dowolnego wierzchołka C , poruszamy się aż do napotkania wierzchołka w pewnej spójnej części G' . Teraz postępujemy indukcyjnie. Obchodzimy stosowny cykl w G' , aż do ewentualnego napotkania kolejnych cykli, obchodzimy je indukcyjnie, po czym wracamy na cykl C w tym samym miejscu, w którym go opuściliśmy, co wynika z faktu, że w należy do cyklu grafu G' . Następnie kontynuujemy wzduż C do napotkania wierzchołka kolejnej części spójnej G' , obchodzimy ją, powracamy na C itd., aż do powrotu do punktu startu. Skończoność grafu gwarantuje skończoność opisanego algorytmu. \square

Powyższy algorytm został podany przez Hierholzera¹.

Idea dowodu Tw. Eulera i wynikająca z niej konstrukcja cyklu Eulera przedstawiona jest przykładowo na rys. 5.9. Najpierw lokalizujemy pewien cykl C w grafie G (pogrubiony kontur na rys. po lewej stronie), co na mocy lematu zawsze możemy zrobić w grafie spójnym o wierzchołkach stopnia parzystego. Następnie usuwamy krawędzie cyklu C i powstałe wierzchołki izolowane z grafu G , w wyniku czego dostajemy graf G' ze środkowej części rysunku. W ogólności ten graf jest niespójny. Natomiast wszystkie jego wierzchołki są stopnia parzystego, zatem w każdej ze spójnych części grafu G' wydzielamy po cyklu, które

¹ Carl Hierholzer (1840-1871), niemiecki matematyk.



Rysunek 5.9: Idea dowodu Tw. Eulera

następnie usuwamy jak wyżej, dostając sytuację z rys. po prawej stronie. Procedurę powtarzamy tak długo, aż nic nie pozostaje z grafu G . Powyższy przepis wydziela z grafu G krawędziowo rozłączne cykle. Etykiety przy krawędziach grafów zaznaczają przykładową kolejność obchodzenia grafu G . Zaczynamy poruszać się wzduż cyklu C po krawędzi 1, następnie wchodzimy na pierwszy napotkany „poboczny” cykl. Jest to dolny cykl ze środkowej części rysunku. Pokonujemy krawędź 2, po czym natrafiamy na cykl z prawej części rysunku. Obchodzimy krawędzie 3, 4, 5, 6, następnie powracamy na cykl ze środkowej części rysunku i obchodzimy krawędzie 7 i 8, po czym powracamy na C . Pokonujemy krawędzie 9 i 10, z kolei obchodzimy górny cykl na środkowym rysunku, krawędzie 11, 12 i 13, powracamy na C i już bez zbaczania kontynuujemy wzduż krawędzi 14, 15 i 16 do punktu wyjścia. W ten sposób znaleźliśmy pewien cykl Eulera w grafie G . Z konstrukcji z dowodu Tw. 5.2 wynikają natychmiast dwa wnioski:

Wniosek 5.1. Skończony graf spójny jest eulerowski wtedy i tylko wtedy, gdy można go podzielić na krawędziowo rozłączne cykle.

Wniosek 5.2. Skończony graf spójny o dokładnie dwóch wierzchołkach stopnia nieparzystego ma drogę Eulera, zaczynającą się w jednym z tych wierzchołków, a kończącą w drugim.

Dowód: Połączmy wierzchołki stopnia nieparzystego dodatkowa krawędzią. W tym momencie graf ma wszystkie wierzchołki stopnia parzystego, jest więc eulerowski. Zaczniemy cykl Eulera od spaceru po dodanej krawędzi. Następnie usuńmy dodaną krawędź, co pozostawia drogę Eulera. \square

Poniższy algorytm pokazuje, jak skonstruować w inny sposób niż metodą z rys. 5.9 drogę Eulera w grafie spełniającym warunki Tw. (5.2).

Alg. 5.1 (konstrukcja Fleury'ego² drogi Eulera).

² M. Fleury, francuski matematyk z XIX w.

1. *Punkt startowy: jeśli wszystkie wierzchołki są stopnia parzystego, wybieramy dowolny wierzchołek. Jeśli dwa wierzchołki są stopnia nieparzystego, to wybieramy jeden z nich. Oznaczamy wierzchołek startowy jako w , następnie inicjalizujemy ciąg wierzchołków W szukanej drogi jako (w) i ciąg krawędzi K jako ciąg pusty.*
2. *Jeśli z wierzchołka nie wychodzi żadna krawędź, STOP. W i K zawierają kolejno odwiedzone wierzchołki i krawędzie grafu, określające cykl lub drogę Eulera.*
3. *Jeśli z wierzchołka wychodzi dokładnie jedna krawędź k , usuwamy wierzchołek, w którym jesteśmy, idziemy do wierzchołka będącego końcem krawędzi k i nazywamy go w . Następnie usuwamy krawędź k i idziemy do kroku 5.*
4. *Jeśli z wierzchołka wychodzi więcej niż jedna krawędź, wybieramy do dalszej drogi taką krawędź k , po której usunięciu graf pozostaje spójny, przechodzimy po tej krawędzi do nowego wierzchołka, który oznaczamy w , po czym usuwamy krawędź k .*
5. *Dopisujemy w na końcu ciągu W oraz k na końcu ciągu K , po czym wracamy do kroku 2.*

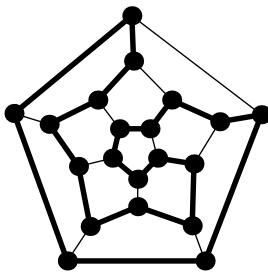
Krok 4 wymaga sprawdzenia spójności grafu, co jest czasochłonne, a czego nauczymy się później (podrozdział 5.7). Czyni to algorytm Fleury'ego znacznie mniej wydajnym od algorytmu Hierholzera. Oznaczmy liczbę krawędzi grafu G jako E . Można pokazać, że złożoność (czas wykonywania) algorytmu Hierholzera rośnie jak E , a algorytmu Fleury'ego jak E^2 .

5.2 Grafy Hamiltona

Zagadnieniem bardzo podobnym w sformułowaniu do problemu Eulera jest problem cyklu i drogi Hamiltona³.

Def. 5.20. *Cyklem Hamiltona w grafie nazywamy cykl, który przechodzi przez każdy wierzchołek dokładnie raz. Podobnie, drogą Hamiltona nazywamy drogę otwartą przechodzącą przez każdy wierzchołek dokładnie raz. Graf posiadający cykl Hamiltona nazywamy hamiltonowskim. Graf posiadający drogę Hamiltona, ale nie posiadający cyklu Hamiltona, nazywamy półhamiltonowskim.*

³ William Rowan Hamilton (1805-1865), irlandzki fizyk, matematyk i astronom.



Powyższy graf ukazuje przykład grafu hamiltonowskiego z wytluszczenym cyklem Hamiltona. Chociaż różnica między zagadnieniem istnienia cyklu Eulera i cyklu Hamiltona polega „jedynie” na zamianie rolami krawędzi i wierzchołków, ma to niezwykle istotne znaczenie. Tak jak w pierwszym przypadku mamy bardzo proste kryterium 5.2 dostarczające warunku koniecznego i dostatecznego dla eulerowskości grafu, a także szybkie algorytmy konstrukcji rozwiązania, tak dla grafów hamiltonowskich analogiczne kryterium nie ma⁴.

Co zatem wiemy o grafach hamiltonowskich? Większość twierdzeń ma postać: „jeśli graf ma dostatecznie dużo krawędzi, to jest hamiltonowski”. Wprowadźmy oznaczenie $s(w)$ na stopień wierzchołka w . A oto najważniejsze twierdzenia:

Tw. 5.3 (Dirac⁵). *Graf prosty G o $n \geq 3$ wierzchołkach i stopniu każdego wierzchołka $s(w) \geq n/2$ jest hamiltonowski.*

Silniejsze twierdzenie ma postać:

Tw. 5.4 (Ore⁶). *Graf prosty G o $n \geq 3$ wierzchołkach i spełniający warunek $s(w_1) + s(w_2) \geq n$ dla każdej pary niesąsiednich wierzchołków jest hamiltonowski.*

Jeszcze silniejsze twierdzenie, które udowodnimy, pochodzi z całkiem niedawnych czasów:

Tw. 5.5 (Bondy⁷, Chvátal⁸ (1972)). *Rozważmy graf prosty G o $n \geq 3$ wierzchołkach, posiadający dwa niesąsiednie wierzchołki u i v o stopniach $s(u) + s(v) \geq n$, oraz graf G' skonstruowany z G poprzez dodanie krawędzi uv . Wówczas G jest hamiltonowski wtedy i tylko wtedy, gdy G' jest hamiltonowski.*

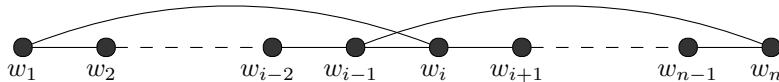
⁴ Uwaga dla tych, którzy znają klasyfikację algorytmów: sprawdzenie, czy dany graf jest hamiltonowski jest problemem trudnym, o klasie złożoności obliczeniowej NPC – *Nondeterministic Polynomial Complete* (patrz rozdz. 6.6).

⁵ Gabriel Andrew Dirac (1925-1984), angielski matematyk, przyczynił się do rozwoju teorii grafów.

⁶ Oystein Ore (1899-1968), norweski matematyk, przyczynił się do rozwoju kombinatoryki i teorii pierścieni.

⁷ John Adrian Bondy (1944-), brytyjsko-kanadyjski matematyk.

⁸ Václav Chvátal (1946-), czesko-kanadyjski matematyk.



Rysunek 5.10: Ilustracja do dowodu Tw. Bondy'ego-Chvátala

Dowód: W oczywisty sposób, jeśli G jest hamiltonowski, to G' jest hamiltonowski. Dowód w drugą stronę przebiega przez sprzeczność. Załóżmy, że G' jest hamiltonowski, a G nie jest. Graf G posiada drogę Hamiltona o początku w $u = w_1$ i końcu w $v = w_n$, która pochodzi z cyklu Hamiltona w G' poprzez usunięcie krawędzi uv . Oznaczmy tę drogę jako $w_1, w_2, \dots, w_{n-1}, w_n$. Teraz kluczowe spostrzeżenie: jeśli wierzchołek w_i , $i = 3, \dots, n-1$ sąsiaduje z w_1 , to wierzchołek w_{i-1} nie może sąsiadować z w_n . W przeciwnym wypadku bowiem mielibyśmy sytuację z rys. 5.10, tzn. moglibyśmy utworzyć cykl Hamiltona $w_1 w_2 \dots w_{i-1} w_n w_{n-1} \dots w_i w_1$. A priori, maksymalny możliwy stopień wierzchołka w_n wynosi $n-2$ (ponieważ nie sąsiaduje z w_1 , może sąsiadować z w_2, \dots, w_{n-1}), czyli $s(w_n) \leq n-2$. Jeśli jednak w_1 sąsiaduje z p wierzchołkami spośród w_3, \dots, w_{n-1} , to na mocy „kluczowego spostrzeżenia” $s(w_n) \leq n-2-p$. Jednocześnie $s(w_1) = p+1$ (1 pochodzi z sąsiadowania z w_2). Tak więc $s(w_1) + s(w_n) \leq n-1$, co przeczy założeniu $s(w_1) + s(w_n) \geq n$. \square

Uwaga 5.1. Tw. 5.4 wynika z Tw. 5.5. Jeśli bowiem połączymy wszystkie niesąsiednie pary wierzchołków, które z założenia spełniają $s(u)+s(v) \geq n$, otrzymamy graf pełny, który w oczywisty sposób jest hamiltonowski.

Uwaga 5.2. Tw. 5.3 wynika natychmiast z Tw. 5.4, ponieważ jeśli stopień każdego wierzchołka jest nie mniejszy od $n/2$, to dla każdej pary, również rozłącznej, suma stopni jest nie mniejsza od n .

Inne twierdzenie, które można dowieść za pomocą Tw. 5.4 jest następujące:

Tw. 5.6. Graf prosty o $n \geq 3$ wierzchołkach jest hamiltonowski, jeżeli ma co najmniej $(n-1)(n-2)/2 + 2$ krawędzi.

Zauważmy, że wszystkie powyższe twierdzenia są słabe, tj. nałożone warunki konieczne są silne. Istnieją bowiem grafy hamiltonowskie mające znacznie mniej krawędzi czy też dużo mniejszy stopień wierzchołków, niż wymagają Tw. 5.3 i 5.4. W szczególności graf cykliczny ukazany na rys. 5.11 ma najmniejszą możliwą liczbę krawędzi (tyle samo, co wierzchołków) wymaganą dla posiadania cyklu. Co gorsza, *niekonstrukcyjne* dowody powyższych twierdzeń nie wskazują na żaden rozsądnie szybki algorytm znajdowania cyklu Hamiltona w grafie, o którym wiemy na mocy twierdzeń, że jest hamiltonowski. Możemy oczywiście

rozpatrzyć wszystkie $n!$ możliwych sekwencji wierzchołków (algorytm brutalnej siły) i sprawdzić, która jest cyklem w grafie, ale złożoność takiego algorytmu nie pozwala na skuteczne zastosowanie go dla większych n . Istnieją nieco lepsze algorytmy, tzw. *algorytmy z powracaniem* (zob. podrozdział 5.12), w których czas znajdowania cyklu Hamiltona w grafie hamiltonowskim rośnie wykładniczo z liczbą wierzchołków.

Na koniec tego podrozdziału nieco rozrywki. Na rys. 5.12 przedstawiony jest cykl Hamiltona dla ruchu konika szachowego po szachownicy 8×8 . Wierzchołki grafu odpowiadają polom szachownicy, a krawędzie dozwolonym ruchom konika „dwa pola do przodu, jedno w bok” (zob. zadanie 5.6).

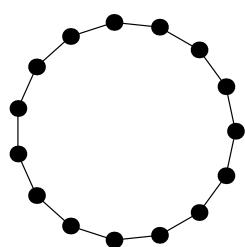
5.3 Kalejdoskop grafów

A teraz kilka dalszych definicji często występujących rodzajów grafów.

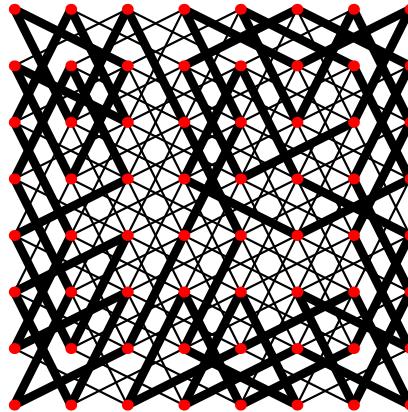
Def. 5.21. *Grafem dwudzielnym nazywamy graf, w którym zbiór wierzchołków można podzielić na dwa niepuste rozłączne podzbiory W_1 i W_2 tak, że każda z krawędzi grafu łączy wierzchołek z W_1 z wierzchołkiem z W_2 (tj. nie ma połączeń w obrębie W_1 ani W_2). Graf dwudzienny pełny to graf dwudzienny prosty, w którym każdy wierzchołek W_1 jest połączony z każdym wierzchołkiem W_2 oraz każdy wierzchołek W_2 jest połączony z każdym wierzchołkiem W_1 . Graf dwudzienny pełny o liczbach wierzchołków n w W_1 i m w W_2 oznaczamy jako $K_{n,m}$.*

Rys. 5.13 pokazuje przykłady grafów dwudzielnych.

Kolejną szacowną rodziną grafów są tzw. *grafy platońskie*, będące grafami tworzonymi przez wierzchołki i krawędzie wielościanów foremnych. Wielościany te nazywane są też *bryłami platońskimi*, o których więcej mówimy w podrozdziale 5.5. Grafy te przedstawione są na rys. 5.14. Rysowanie tych grafów na płaszczyźnie można sobie wyobrazić w taki sposób: bierzemy wielościan z elastycznymi krawędziami i deformujemy go, rozszerzając jedną z jego ścian na tyle, że jej krawędzie znajdują się na zewnątrz pozostałe części grafu. Po zrzutowaniu na płaszczyznę krawędzie grafu nie przecinają się i uzyskujemy rezultat jak na rys. 5.14.



Rysunek 5.11: Graf cykliczny



Rysunek 5.12: Cykl Hamiltona dla ruchu konika szachowego po szachownicy o wymiarach 8×8 . Wierzchołki odpowiadają środkom pól szachownicy

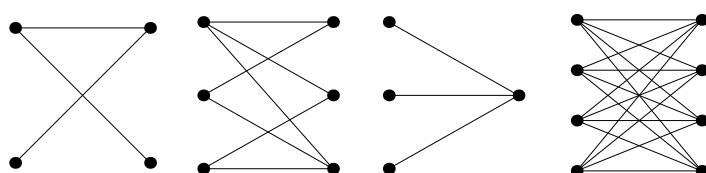
Wreszcie niezwykle ważna, również ze względu na zastosowania w informatyce, rodzina grafów, mianowicie *drzewa*.

Def. 5.22. Drzewem nazywamy graf spójny prosty niezawierający cykli. Las to graf niespójny, którego spójne części są drzewami. Wierzchołki stopnia 1 w drzewie nazywamy wierzchołkami wiszącymi lub liśmi. Krawędzie drzewa nazywamy gałęziami.

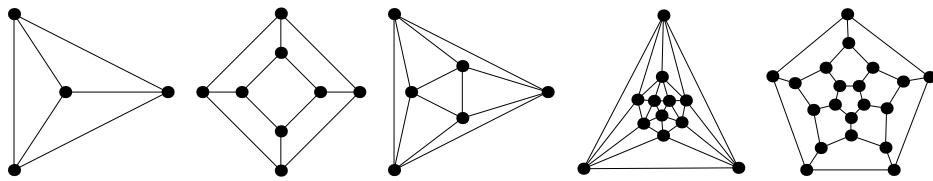
Przykładowe drzewa pokazane są na rys. 5.15. Razem stanowią przykład lasu o trzech drzewach.

Tw. 5.7. Liczba krawędzi drzewa o n wierzchołkach wynosi $n - 1$.

Dowód: (przez indukcję) Twierdzenie w oczywisty sposób zachodzi dla $n = 1$, gdzie liczba krawędzi wynosi 0. Dodanie kolejnego wierzchołka w drzewie związane jest z dodaniem dokładnie jednej krawędzi. Istotnie, możemy dodać wierzchołek na istniejącej krawędzi, wtedy liczba krawędzi rośnie o 1, lub też poza istniejącymi krawędziami, kiedy to dodajemy krawędź między nowym wierzchołkiem a istniejącym wierzchołkiem. \square



Rysunek 5.13: Grafy dwudzielne. Dwa grafy po prawej stronie są dwudzielne pełne



Rysunek 5.14: Grafy platońskie dla czworościanu, sześciangu, ośmiościanu foremnego, dwudziestościanu foremnego i dwunastościanu foremnego

Zauważmy, że jest to najmniejsza możliwa liczba krawędzi w grafie spójnym o n wierzchołkach. Odjęcie jednej krawędzi od drzewa (patrz rys. 5.15) powoduje jego rozpadnięcie się na dwie rozłączne części. Właśnie z tego faktu minimalnej ilości krawędzi wynika znaczenie drzew – możemy wszędzie się dostać, a nie marnujemy materiału na zbędne połączenia!

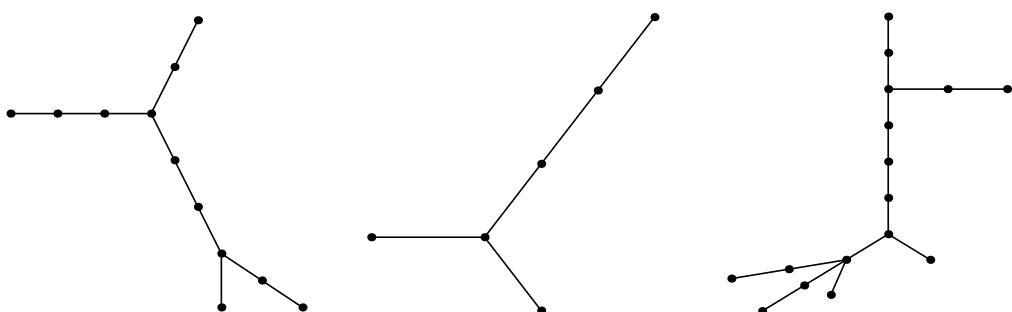
Tw. 5.8. *Następujące stwierdzenia są równoważne:*

1. D jest drzewem.
2. Dowolne dwa wierzchołki D połączone są tylko jedną drogą.
3. Dodanie dowolnej nowej krawędzi do D tworzy dokładnie jeden cykl

Twierdzenie jest intuicyjnie oczywiste. Więcej o drzewach i ich zastosowaniach powiemy w rozdz. 5.16.

5.4 Izomorfizm grafów

Rozważmy grafy rys. 5.16. Widzimy, że są one właściwie takie same jeśli chodzi o połączenia. Wyobraźmy sobie, że krawędzie są elastyczne. Jeśli złapiemy graf z prawej strony rysunku za prawą krawędź i obróćmy ją o pół pełnego obrotu wzduł poziomej osi leżącej na płaszczyźnie rysunku, to „rozplączemy” graf i dostaniemy graf z lewej strony rysunku. Mówiąc bardziej precyzyjnie, jeśli zrobimy



Rysunek 5.15: Przykłady drzew

odwzorowanie

$$\begin{aligned} a' &\rightarrow a, b' \rightarrow b, c' \rightarrow d, d' \rightarrow c \\ 1' &\rightarrow 1, 2' \rightarrow 2, 3' \rightarrow 3, 4' \rightarrow 4, \end{aligned}$$

to funkcja γ dla grafu po prawej stronie przechodzi w funkcję γ dla grafu po lewej stronie. Istotnie,

$$\begin{aligned} \gamma(1') &= \{a', b'\} \rightarrow \gamma(1) = \{a, b\} \\ \gamma(2') &= \{b', d'\} \rightarrow \gamma(2) = \{b, c\} \\ \gamma(3') &= \{c', d'\} \rightarrow \gamma(3) = \{c, d\} \\ \gamma(4') &= \{a', c'\} \rightarrow \gamma(4) = \{a, d\} \end{aligned}$$

Matematycznie, tę własność grafów nazywamy *izomorfizmem*.

Def. 5.23. *Grafy G i H są izomorficzne, jeśli istnieje takie wzajemnie jednoznaczne przekształcenie f między wierzchołkami G i H oraz krawędziami G i H , że liczba krawędzi łącząca dwa dowolne wierzchołki grafu G , w_1 i w_2 , jest równa liczbie krawędzi łączących wierzchołki $f(w_1)$ i $f(w_2)$ w grafie H .*

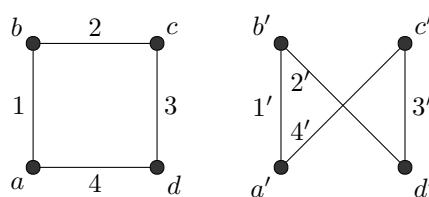
Powyższy przykład pokazuje, że ten sam graf możemy graficznie, czyli w postaci rysunku na płaszczyźnie, reprezentować na bardzo wiele różnych sposobów. Dla większej liczby wierzchołków stwierdzenie, czy dwa grafy są izomorficzne, wcale nie jest oczywiste i trzeba zastosować odpowiedni algorytm porównywania. Ilustrujemy ten fakt na rys. 5.17, gdzie wszystkie grafy są izomorficzne do tzw. grafu Petersena, pokazanego na końcu, a na pierwszy rzut oka wyglądają bardzo różnie.

Pokazany tu graf Petersena ma ciekawą własność. Jest to przykład tzw. *grafo regularnego*, w tym przypadku o stopniu wierzchołków wynoszącym 3.

Def. 5.24. *Graf regularny to graf, w którym wszystkie wierzchołki mają taki sam stopień.*

Inne przykłady grafów regularnych przedstawione są na rys. 5.18

A teraz kolejna definicja dotycząca „podobieństwa” grafów.



Rysunek 5.16: Grafy izomorficzne

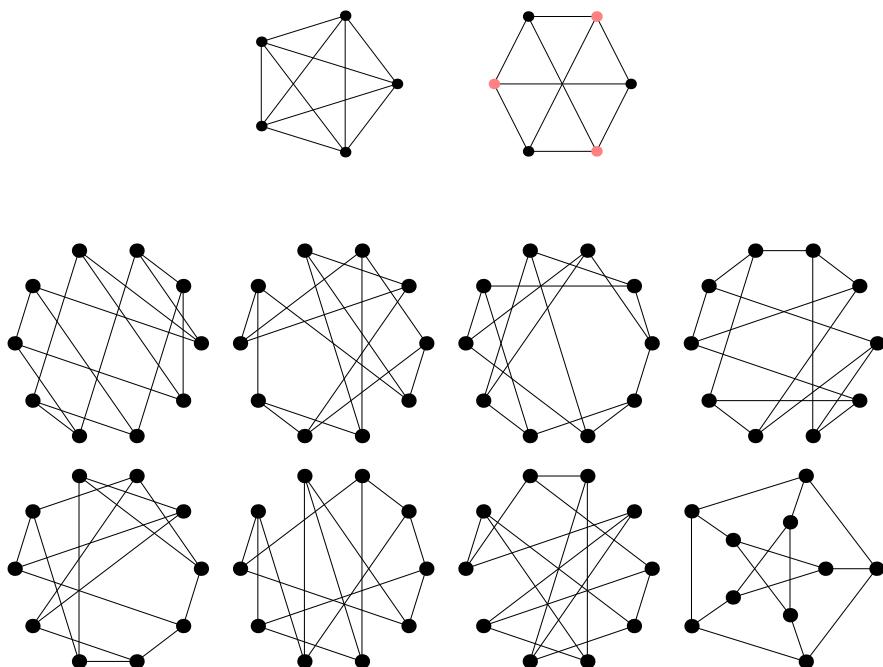
Def. 5.25. Grafy G i G' są homeomorficzne, jeśli jeden może być uzyskany z drugiego poprzez dodanie lub usunięcie na krawędziach dowolnej liczby wierzchołków stopnia 2.

Przykład grafów homeomorficznych pokazany jest na rys. 5.19. Innymi słowy, grafy te mają „dodatkowe” wierzchołki o stopniu 2 „nanizane” na krawędzie. Jest intuicyjnie oczywiste, że grafy homeomorficzne są podobne, chociaż mająną różną liczbę wierzchołków. Ta cecha będzie istotna przy klasyfikacji grafów planarnych w rozdz. 5.5.

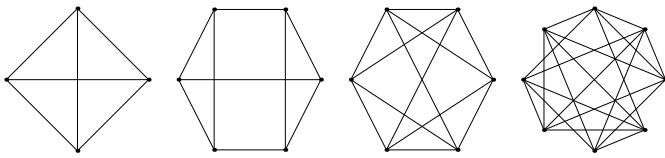
5.5 Grafy planarne

Def. 5.26. Grafem planarnym (płaskim, płaszczyznowym) jest graf, który można narysować na płaszczyźnie w taki sposób, że krawędzie się nie przecinają.

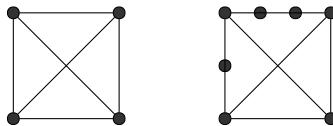
Przykład pokazany jest na rys. 5.20, przedstawiającym ten sam graf narysowany na dwa różne sposoby. Graf po prawej stronie, ewidentnie narysowany na płaszczyźnie, powstał z rysunku po lewej stronie poprzez deformację jednej z krawędzi. Oczywiście takie rozciąganie nie zmienia samego grafu, tylko jego reprezentację graficzną. Grafami nieplanarnymi o podstawowym znaczeniu są graf pełny K_5 i graf dwudzielny pełny $K_{3,3}$:



Rysunek 5.17: Grafy izomorficzne do grafu Petersena



Rysunek 5.18: Przykładowe grafy regularne stopni, odpowiednio, 3, 3, 4 i 5



Rysunek 5.19: Grafy homeomorficzne

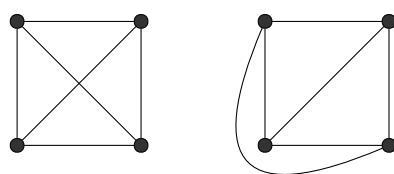
Można się łatwo przekonać, że żadne rozciąganie i rozplątywanie krawędzi nie powiedzie się i grafów tych nie można narysować na płaszczyźnie tak, aby krawędzie się nie przecinały. Tym samym, grafy te nie są planarne.

W 1930 r. Kuratowski⁹ dowódł słynne kryterium planarności grafu.

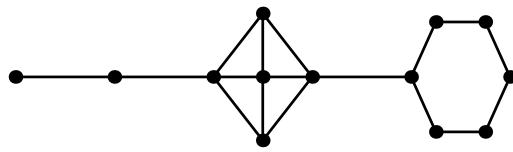
Tw. 5.9 (Kuratowskiego). *Graf jest planarny wtedy i tylko wtedy, gdy nie zawiera podgrafa homeomorficznego z K_5 lub $K_{3,3}$.*

Dowód twierdzenia w stronę \Rightarrow jest natychmiastowy. Twierdzenie przeciwstawne mówi bowiem, że jeśli graf zawiera podgraf homeomorficzny z K_5 lub $K_{3,3}$, to nie jest planarny. Ponieważ K_5 i $K_{3,3}$ nie są planarne, graf je zawierający, posiadający większą liczbę krawędzi i wierzchołków, „tym bardziej” nie może być planarny. Skomplikowany dowód w drugą stronę (\Leftarrow) pomijamy. Pragniemy natomiast zwrócić uwagę na prostotę i zarazem głębię tego twierdzenia. Wystarczy dokonać inspekcji grafu i sprawdzić, czy zawiera on łatwe do rozpoznania grafy homeomorficzne do grafu pełnego o pięciu wierzchołkach, K_5 , lub do grafu dwudzielnego pełnego $K_{3,3}$.

⁹ Kazimierz Kuratowski (1896-1980), polski matematyk o wielkich zasługach w rozwoju teorii mnogości i topologii.



Rysunek 5.20: Graf planarny. Po rozciągnięciu jednej krawędzi rysunku po lewej stronie graf daje się narysować na płaszczyźnie



Rysunek 5.21: Inny przykład grafu planarnego

Podamy teraz bardzo podstawowe fakty dotyczące topologicznych własności grafów. Najpierw zauważmy, że każdy spójny graf planarny w ogólności składa się z połączonych części jednospójnych, 2-spójnych i 3-spójnych. Przykład z rys. 5.21 posiada część 3-spójną w swojej środkowej części, połączoną jedną krawędzią z częścią 2-spójną po prawej stronie oraz z doczepioną częścią jednospójną po lewej stronie. Gdyby graf posiadał tylko część 3-spójną, można by na nią popatrzeć jak na wielościan (w naszym przykładzie byłaby to piramida o podstawie czworokąta widziana od góry, z podstawą identyfikowaną z zewnętrznym obszarem). Stąd właśnie nazwa podanego niżej twierdzenia.

Rysowanie grafów na płaszczyźnie jest równoważne rysowaniu ich na powierzchni kuli. Jest to oczywiste, bo cokolwiek możemy narysować na kartce papieru, możemy też narysować na globusie. Tak więc grafy planarne możemy też określić jako możliwe do narysowania na sferze.

Def. 5.27. *Rozłączne obszary, na które krawędzie grafu planarnego dzielą sferę, na której graf jest narysowany, nazywamy ścianami.*

Euler udowodnił bardzo ogólne twierdzenie dotyczące grafów planarnych:

Tw. 5.10 (wzór wielościanu Eulera). *W grafie planarnym spójnym liczba wierzchołków w , krawędzi k i ścian s spełnia związek $w - k + s = 2$.*

Zauważmy, że w przykładzie z rys. 5.21 $w = 13$, $k = 17$ i $s = 6$ i rzeczywiście $13 - 17 + 6 = 2$.

Dowód: Dowód będzie przebiegał przez indukcję względem dodawania krawędzi i wierzchołków. Twierdzenie w oczywisty sposób zachodzi dla najprostszego możliwego grafu, czyli N_1 . Mamy jeden wierzchołek, zero krawędzi oraz jedną ścianę, gdyż sfera, na której narysowano jeden punkt, nie została podzielona na rozłączne obszary. Następnie rozważmy graf G , dla którego zachodzi teza twierdzenia. Możemy utworzyć inny planarny graf spójny, dodając do G elementy w następujący sposób:

1. Wierzchołek na istniejącej krawędzi: $w \rightarrow w + 1$, $s \rightarrow s$, $k \rightarrow k + 1$.
2. Krawędź między istniejącymi wierzchołkami: $w \rightarrow w$, $s \rightarrow s + 1$, $\rightarrow k + 1$.
3. Nowy wierzchołek izolowany i krawędź między nim a innym istniejącym wierzchołkiem G : $w \rightarrow w + 1$, $s \rightarrow s$, $k \rightarrow k + 1$.

4. Pętla wokół wierzchołka: $w \rightarrow w$, $s \rightarrow s + 1$, $k \rightarrow k + 1$.

Wszystkie te sytuacje przedstawione są na rys. 5.22. Łatwo się przekonać, że istotnie są to wszystkie możliwości skonstruowania nowego spójnego grafu planarnego poprzez dodanie wierzchołka lub krawędzi do grafu G . Widzimy też, że w każdym przypadku nie zmienia się wartość wyrażenia $w - k + s$, które jest *niezmienikiem topologicznym* dla grafów rysowanych na sferze. \square

A teraz ważny wniosek wynikający z powyższego twierdzenia dotyczący wielościanów foremnych, tj. brył platońskich. Podamy najpierw definicję w języku teorii grafów, gdyż w naszej analizie identyfikujemy wierzchołki i krawędzie bryły z wierzchołkami i krawędziami grafu.

Def. 5.28. *Wielościanem foremnym nazywamy spójny prosty graf planarny, który jest m -regularny, przy czym $m \geq 3$, a każda jego ściana ma p krawędzi, gdzie $p \geq 3$.*

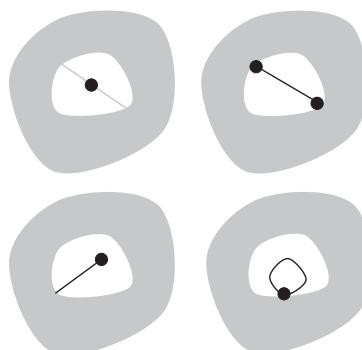
W świetle wcześniejszych definicji oznacza to, że graf ten nie ma pętli ani krawędzi wielokrotnych, stopień każdego wierzchołka, oznaczony jako m , jest jednakowy oraz każda ściana ma jednakową liczbę krawędzi. Oczywiste graniczenie $m \geq 3$ wynika z faktu, że pragniemy rozważać wielościany, gdzie przynajmniej trzy krawędzie łączą się w jednym wierzchołku. Podobnie, każda ściana musi mieć przynajmniej trzy krawędzie. Przykładem wielościanu foremnego jest graf powstały z wierzchołków i krawędzi sześcianu, gdzie $m = 3$ i $p = 4$.

Zgodnie z lematem o uściskach dloni (5.1) mamy związek

$$wm = 2k. \quad (5.7)$$

Dodatkowo, ponieważ każda ściana ma p krawędzi, otrzymujemy równość

$$ps = 2k. \quad (5.8)$$



Rysunek 5.22: Przypadki do rozpatrzenia w dowodzie Tw. 5.10. Szary obszar wok. oznacza elementy grafu, które nie ulegają zmianie w kroku indukcyjnym

Tabela 5.1: Podstawowe własności wielościanów foremnych (brył platońskich)

Stopień wierzchołka m	Liczba krawędzi ściany p	Liczba ścian s
3	3	4
3	4	6
3	5	12
4	3	8
5	3	20

Czynnik 2 wynika z faktu, że każda krawędź przylega do dwóch ścian, tym samym zliczając krawędzie po wszystkich ścianach, uwzględniamy je podwójnie. Tak więc $k = ps/2$ oraz $wm = 2k = ps$. Korzystając z równości we wzorze z Tw. (5.10), znajdujemy następujący związek między liczbą ścian s , stopniem wierzchołka m i liczbą krawędzi ściany p :

$$\frac{ps}{m} - \frac{ps}{2} + s = 2. \quad (5.9)$$

Obliczając s , dostajemy

$$s = \frac{4m}{2p - pm + 2m} = \frac{4m}{4 - (p - 2)(m - 2)}. \quad (5.10)$$

W oczywisty sposób $s > 0$, zatem $(p - 2)(m - 2) < 4$. Ponieważ $p \geq 3$ i $m \geq 3$, wynika stąd w szczególności, że $m < 6$. Dalsza analiza wykorzystuje fakt, że liczba ścian s musi być liczbą naturalną. Rozważmy równanie (5.10) dla kolejnych dopuszczalnych wartości m , zaczynając od $m = 3$. Mamy wówczas

$$s = \frac{12}{6 - p},$$

zatem s jest naturalne dla $p = 3$ (wtedy $s = 4$ – czworościan), $p = 4$ ($s = 6$ – sześcian) i $p = 5$ ($s = 12$ – dwunastościan foremny). Dla $m = 4$

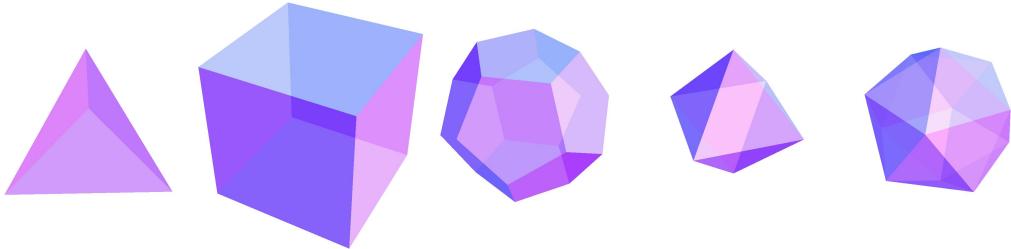
$$s = \frac{8}{4 - p},$$

co daje $p = 3$ ($s = 8$ – ośmiościan). Wreszcie dla $m = 5$

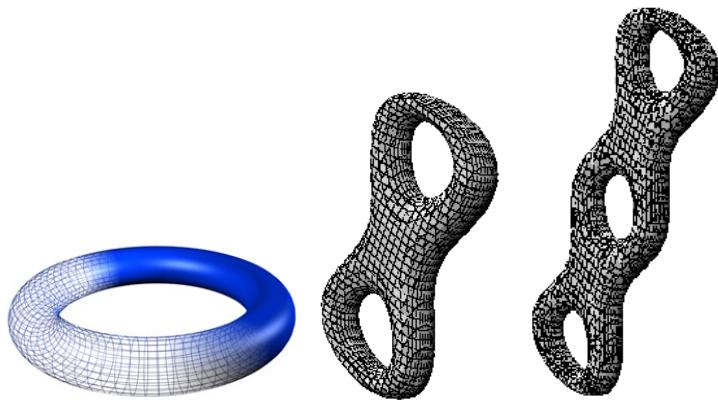
$$s = \frac{20}{10 - 3p},$$

skąd $p = 3$ ($s = 20$ – dwudziestościan foremny). Wszystkie przypadki zebrane są w tabeli 5.1. Rysunek 5.23 przedstawia wszystkie bryły platońskie.

A teraz wyobraźmy sobie inną powierzchnię niż sfera, mianowicie torus (zob. rys. 5.24).



Rysunek 5.23: Bryły platońskie

Rysunek 5.24: Powierzchnie o różnych topologiach: torus, 2-torus, 3-torus. Tworzenie tych powierzchni można sobie wyobrażać jako „dziurawienie” sfery lub jako doklejanie do niej uchwytów czy rączek. Liczba g tych dziur (lub rączek) określa charakterystykę χ Eulera

Tw. 5.11 (wzór wielościanu Eulera dla dowolnej topologii). *W grafie, do którego narysowania bez przecinania krawędzi potrzebna jest powierzchnia z g rączkami, liczba wierzchołków w , krawędzi k i ścian s spełnia związek*

$$w - k + s = 2 - 2g.$$

Dowód: Wyobraźmy sobie, że w celu narysowania grafu musimy dobudować rączkę (zob. rys. 5.24). Rączka utożsamia dwie ściany które łączy, zatem liczba ścian maleje o 1. Następnie musimy na rączce narysować krawędź, gdyż właśnie po to dodawaliśmy rączkę! A więc liczba krawędzi rośnie o 1. Tym samym niezmiennik topologiczny $w - k + s$ maleje o 2 w stosunku do grafu planarnego, tj. $w - k + s = 0$. Teraz możemy indukcyjnie dobudowywać graf, rysując dalsze krawędzie na istniejącej powierzchni z rączką, co nie zmienia wartości niezmiennika, podobnie jak w dowodzie Tw. 5.10. Dobudowanie każdej kolejnej rączki i krawędzi powoduje zmniejszenie niezmiennika o 2, skąd wynika teza. \square

Tabela 5.2: Liczba różnych typów grafów prostych nieizomorficznych o n wierzchołkach [45]

Typ grafów prostych	n=2	3	4	5	6	7	8	9
wszystkie	2	4	11	34	156	1044	12346	274668
spójne	1	2	6	21	112	853	11117	261080
eulerowskie	0	1	1	4	8	37	184	1782
hamiltonowskie	0	1	3	8	48	383	8196	177083
spójne planarne	1	2	6	20	99	646	5974	71885

5.6 Ile jest grafów?

Zliczanie grafów jest jednym z ważniejszych problemów kombinatorycznych. Zazwyczaj sprowadza się do zliczania typów grafów nieizomorficznych posiadających daną własność, która jest istotna dla rozważanego zagadnienia.

Policzmy najpierw, ile jest wszystkich grafów *prostych* o n wierzchołkach i dowolnej liczbie krawędzi, bez żadnych dodatkowych warunków ani bez utożsamiania grafów izomorficznych czy homeomorficznych. Warunek ograniczenia się do grafów prostych jest tu oczywiście istotny, bo np. dopuszczenie dowolnie wielu krawędzi wielokrotnych dałoby nieskończoność. A więc liczymy wszystkie możliwe grafy proste. Pamiętamy, że jeśli dany zbiór skończony ma $|A|$ elementów, to możemy zatem zeń utworzyć $2^{|A|}$ różnych podzbiorów, ponieważ każdy element może należeć do podzbioru lub nie należeć. Na przykład podzbiorami zbioru trójelementowego $A = \{a, b, c\}$ jest $2^3 = 8$ podzbiorów: zbiór pusty \emptyset , zbiory jednoelementowe $\{a\}, \{b\}, \{c\}$, zbiory dwuelementowe $\{a, b\}, \{b, c\}, \{a, c\}$ oraz zbiór trójelementowy $\{a, b, c\}$. Ponieważ zbiór możliwych krawędzi w grafie o n wierzchołkach jest właśnie podzbiorem zbioru par wierzchołków, których mamy $n(n - 1)/2$, to wynika stąd natychmiast, że mamy $2^{n(n-1)/2}$ różnych grafów *prostych* o n wierzchołkach i dowolnej liczbie krawędzi, spójnych lub niespójnych. Pierwszych kilka wyrazów tego ciągu to

$$1, 2, 8, 64, 1024, 32768, 2097152, 268435456, 68719476736, 35184372088832, \dots$$

wzrost jest nawet szybszy niż $n!$. Jeśli identyfikujemy grafy izomorficzne, wówczas możemy policzyć typy grafów, których jest oczywiście mniej niż wszystkich grafów.

Rozwiązanymi problemami zliczania grafów są np. zliczanie grafów spójnych i eulerowskich zawierających daną liczbę wierzchołków i krawędzi, natomiast nie są rozwiązane problemy zliczania grafów planarnych czy hamiltonowskich. Tabela 5.2 przedstawia liczby typów grafów nieizomorficznych o n wierzchołkach dla początkowych wartości n . Rzeczą jasna, nakładanie warunków zmniejsza, i to bardzo istotnie, liczbę grafów. Problem znajdowania niektórych rodzajów grafów jest bardzo trudnym zadaniem. Na przykład dla grafów hamiltonowskich i grafów spójnych planarnych wartości tabeli 5.2 znane są, przy dzisiejszych komputerach (rok 2021), tylko do $n = 12$.

5.7 Przeszukiwanie grafów

Bardzo ważnym problemem o praktycznych zastosowaniach jest tzw. *przeszukiwanie* grafów. Chodzi o to, aby umieć dotrzeć do *każdego* wierzchołka danego grafu, tzn. znaleźć prowadzącą do niego drogę. Wyobraźmy więc sobie, że niczym Tezeusz¹⁰ stojimy przy wejściu do labiryntu i mamy go zbadać. Przedstawimy dwa podstawowe algorytmy przeszukiwania grafów, tzw. przeszukiwanie *w głąb* i *wszerz*. Algorytmy te są często częściami składowymi innych, bardziej rozbudowanych algorytmów działających na grafach, dlatego ich poznanie i zrozumienie jest szczególnie ważne.

Przykładowe działanie algorytmu przeszukiwania w głąb przedstawione jest na rys. 5.25. Punkt startowy, czyli wejście do labiryntu, oznaczony jest grubą kropką na lewym górnym rysunku. Niech wierzchołek startowy ma numer 1, a pozostałe kolejne numery od 2 do 8, w kolejności odwrotnej do ruchu wskazówek zegara. Na starcie mamy do wyboru dwa korytarze idące do wierzchołków 4 i 5. Wybieramy dowolny z nich i podążamy do komory na jego końcu (4), którą oznaczamy kropką. Oznaczamy też pogrubieniem korytarze, które doprowadziły nas do nowych komór. Z tego miejsca idziemy dalej do nieodwiedzonej jeszcze komory (2), z niej do kolejnej (3) itd., za każdym razem znakując odwiedzone wierzchołki. Istotą algorytmu jest więc podążanie jak najdalej bez wracania. W ten sposób odnajdujemy komory 6, 7 i 5. Cofamy się dopiero wtedy, gdy dalsze podążanie „w głąb” nie prowadzi już do nieodwiedzonego jeszcze wierzchołka. W naszym przykładzie ma to miejsce dopiero na siódmym rysunku, kiedy to musimy pójść wstecz wzdułż dwóch krawędzi do komory 5, aby móc wybrać korytarz prowadzący do ostatniej komory 8. Na tym kroku algorytm się kończy, gdyż mamy znalezione połączenia do wszystkich wierzchołków grafu.

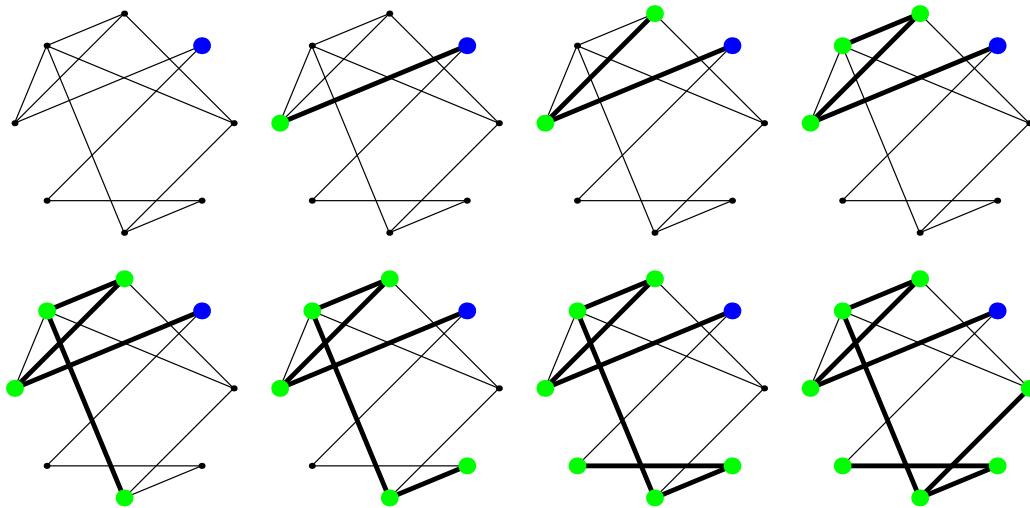
Zauważmy, że w wyniku naszej eksploracji powstało *drzewo spinające* wyjściowego grafu.

Def. 5.29. *Drzewo spinające grafu G to drzewo, które jest podgrafem G i zawiera wszystkie jego wierzchołki.*

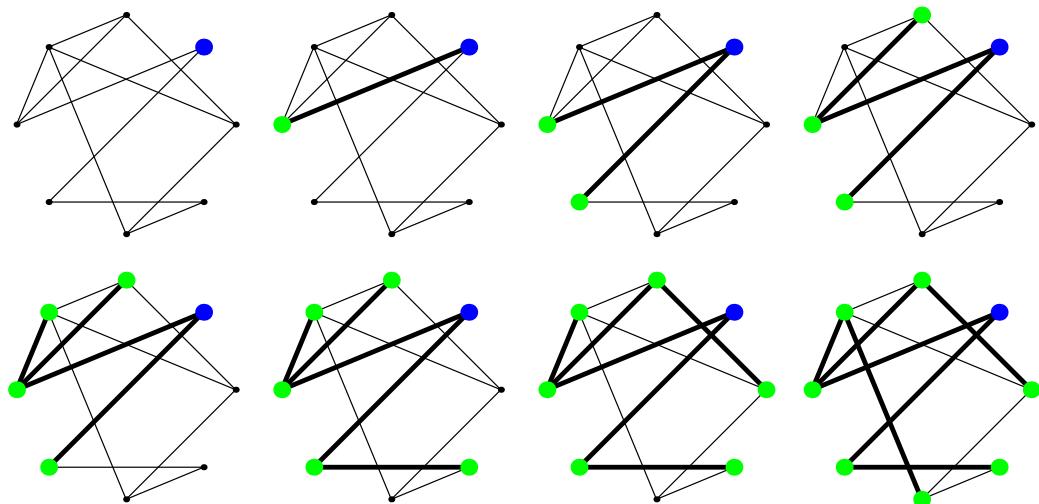
Def. 5.30. *Krawędzie należące do grafu G, a nienależące do danego drzewa spinającego D nazywamy cięciwami G względem drzewa D.*

Odmienna strategia, tzw. przeszukiwanie wszerz, pokazana jest dla tego samego grafu na rys. 5.26. Startujemy z tego samego punktu i pierwszy pokonany korytarz jest ten sam, prowadząc nas do komory 4. Teraz jednak, zamiast pędzić do przodu, wracamy bojaźliwie do punktu startowego i idziemy z tego miejsca drugim korytarzem do innej nieodwiedzonej komory, 5. W tym momencie (trzeci rysunek) mamy znalezione wierzchołki odległe od wierzchołka początkowego o 1

¹⁰ Tezeusz – mityczny bohater, który zaopatrzony w nić Ariadny zdołał wydostać się z labiryntu po zabiciu Minotaury. Wchodząc do labiryntu, rozwijał nić, która później wskazywała drogę powrotną.



Rysunek 5.25: Przykład przeszukiwania grafu w głąb



Rysunek 5.26: Przykład przeszukiwania grafu wszerz

krawędź. Teraz wracamy do pierwszego odwiedzonego wierzchołka (4) i odnajdujemy wszystkie wierzchołki odległe od niego o 1, czyli 2 i 3, z kolei udajemy się do komory 5 i odszukujemy komorę 7. Nowo znalezione komory 2, 3 i 7 są odległe od punktu startu o 2 krawędzie. Teraz wracamy do pierwszego znalezionego wierzchołka odległego od startu o 2 krawędzie i znajdujemy wierzchołek 8, następnie wracamy do wierzchołka 3 i odnajdujemy ostatni wierzchołek 6. Komory 6 i 8 są odległe od startu o 3 krawędzie. W wyniku ponownie powstało drzewo spinające wyjściowego grafu, rzecz jasna inne niż w przypadku przeszukiwania w głąb.

Podajemy teraz obydwa algorytmy bardziej formalnie. Rozważmy spójny graf G i początkowy wierzchołek w_0 .

Alg. 5.2 (przeszukiwanie w głąb).

1. Ustal $w = w_0$ (aktualny wierzchołek) i $n = 0$ (aktualna liczba krawędzi drzewa D). Początkowe drzewo D zawiera jedynie w_0 .
2. Jeśli jest to możliwe, wybierz krawędź e_n przyległą do w i idź nią do nowego (czyli jeszcze niezawartego w D) wierzchołka w_{n+1} . Dodaj e_n i w_{n+1} do D , ustal $w = w_{n+1}$ i zwiększ n o 1.
3. Póki można, powtórz krok 2.
4. Jeżeli D zawiera wszystkie wierzchołki G , zakończ algorytm. D jest drzewem spinającym grafu G o n krawędziach.
5. Jeśli $w_i = w$ nie sąsiaduje już z żadnym nieodwiedzonym wierzchołkiem, cofnij się do w_{i-1} , ustal $w = w_{i-1}$, następnie idź do kroku 2.

Alg. 5.3 (przeszukiwanie wszerz).

1. Ustal $w = w_0$ (aktualne „centrum poszukiwań”), $n = 0$ (aktualna liczba krawędzi D) i $k = 1$. Początkowe drzewo D zawiera jedynie w_0 .
2. Jeśli jest to możliwe, wybierz krawędź e_n przyległą do w i idź nią do nowego (czyli jeszcze niezawartego w D) wierzchołka w_{n+1} . Dodaj e_n i w_{n+1} do D i zwiększ n o 1.
3. Póki można, powtórz krok 2.
4. Jeżeli D zawiera wszystkie wierzchołki G , zakończ algorytm. D jest drzewem spinającym grafu G o n krawędziach.
5. Ustal nowe centrum poszukiwań jako $w = w_k$, zwiększ k o 1 i idź do kroku 2.

Uwaga 5.3. Zauważmy, że algorytmy przeszukiwania wszerz i w głąb można stosować do badania spójności grafu. Jeśli uzyskane drzewo spinające D zawiera wszystkie wierzchołki grafu G , graf ten jest spójny.

Złożoność obliczeniowa obydwu algorytmów wynosi $k+n$, gdzie k jest liczbą krawędzi, a n liczbą wierzchołków przeszukiwanego grafu.

5.8 Grafy z wagami

Dotychczas interesowały nas zwykłe grafy, gdzie istotny był tylko fakt połączenia wierzchołków krawędzią. Pytaliśmy więc, czy jesteśmy w stanie dojechać z Sędziszowa do Augustowa koleją, a nie interesowała nas odległość, czas podróży czy cena biletu. Aby móc modelować zagadnienia, w których cena biletu jednak nas interesuje, wprowadza się pojęcie *grafu z wagami* (krawędziowymi).

Def. 5.31. *Grafem (skierowanym lub nieskierowanym) z wagami krawędziowymi nazywamy graf, w którym wszystkim krawędziom przyporządkowano liczbę rzeczywistą, zwaną wagą krawędzi.*

Waga może być ujemna. Wyobraźmy sobie, że waga jest różnicą wysokości punktu końcowego i początkowego na poszczególnych etapach wycieczki górskiej, która na podejściach jest dodatnia, a na zejściach ujemna. W niektórych sytuacjach (dodatnie wagi, graf nieskierowany, zachowanie nierówności trójkąta) funkcja wag jest metryką w grafie. Jest tak np. dla wag będących odległościami euklidesowymi między miejscowościami. Grafy bez wag rozważane wcześniej można traktować jako grafy z wagą równą 1 dla każdej krawędzi. W ten sposób przedstawione niżej twierdzenia i algorytmy stosują się również do „zwykłych” grafów.

Def. 5.32. *Wagą grafu nazywamy sumę wag jego wszystkich krawędzi.*

Jednym z podstawowych problemów optymalizacyjnych teorii grafów z wagami jest znalezienie tzw. *minimalnego drzewa spinającego*.

Def. 5.33. Minimalne drzewo spinające grafu G to drzewo spinające, którego waga jest mniejsza lub równa wadze innych drzew spinających grafu G .

Inaczej, jest to po prostu „najkrótsza” (choć niekoniecznie jednoznacznie określona) sieć połączeń, która spina graf. Wyobraźmy sobie, że planujemy sieć kolejową tak, aby połączyć wszystkie miejscowości i aby sumaryczna długość torów była jak najmniejsza. Rozwiązaniem jest właśnie minimalne drzewo spinające.

Tablica wag w_{jk} dla grafu prostego z wagami zawiera wagi krawędzi między wierzchołkami j i k . Jeśli wierzchołki są niepołączone, odpowiedni element macierzy wynosi ∞ . Rozważmy graf „kraciasty” z rys. 5.27. Na lewym górnym rysunku oznaczone są wierzchołki a, b, \dots, i . Tablica wag dla tego grafu ma postać tabeli 5.3. Ponieważ graf jest nieskierowany, macierz wag jest symetryczna. W przypadku grafów skierowanych macierz wag w ogólności nie musi być symetryczna.

Poznamy teraz dwa niezwykle proste algorytmy znajdowania minimalnego drzewa spinającego. Pierwszy z nich został podany przez Jarnika¹¹, a następnie ponownie odkryty przez Prima¹², po czym jeszcze raz niezależnie znalazł go Dijkstra!¹³

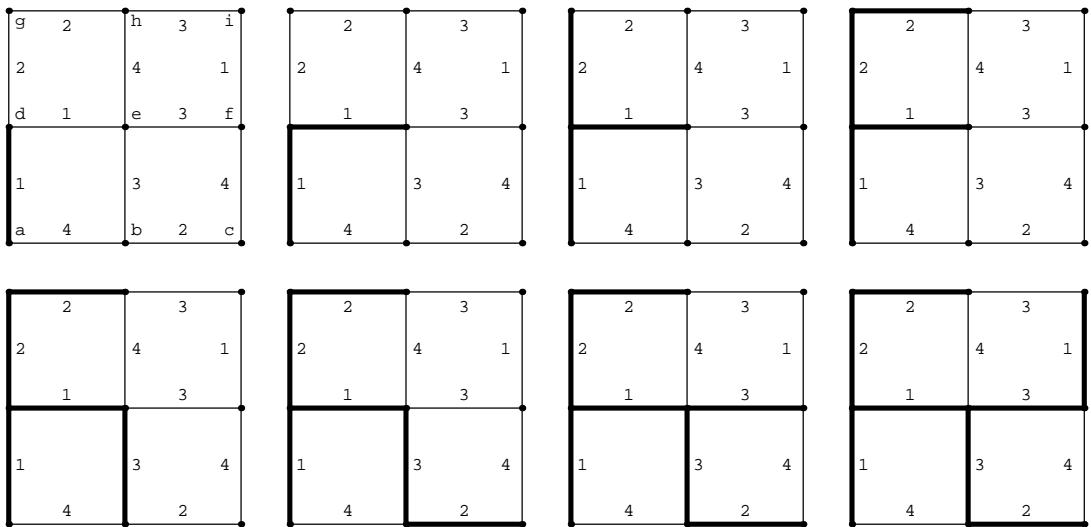
Alg. 5.4 (algorytm Jarnika (Prima) znajdowania minimalnego drzewa spinującego). Niech G będzie grafem spójnym z wagami

1. Utwórz początkowe drzewo D składające się z dowolnego wierzchołka G .
2. Utwórz zbiór wszystkich krawędzi K .

¹¹ Vojtěch Jarník (1897-1970), czeski matematyk.

¹² Robert Clay Prim (1921-), amerykański matematyk i informatyk

¹³ Edsger Dijkstra (1930-2002), holenderski informatyk.



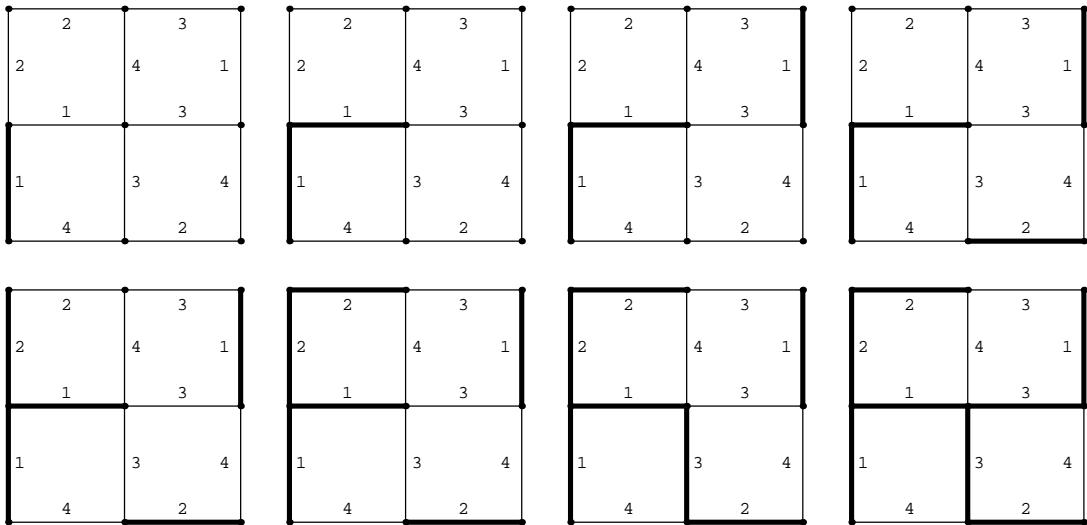
Rysunek 5.27: Kolejne kroki algorytmu Jarnika (Prima). Liczby oznaczają wagę krawędzi. Krawędzie wytłuszczone budują minimalne drzewo spinające

Tabela 5.3: Tablica (macierz) wag dla grafu z rys. 5.27

	a	b	c	d	e	f	g	h	i
a	∞	4	∞	1	∞	∞	∞	∞	∞
b	4	∞	2	∞	3	∞	∞	∞	∞
c	∞	2	∞	∞	∞	4	∞	∞	∞
d	1	∞	∞	∞	1	∞	2	∞	∞
e	∞	3	∞	1	∞	3	∞	4	∞
f	∞	∞	4	∞	3	∞	∞	∞	1
g	∞	∞	∞	2	∞	∞	∞	2	∞
h	∞	∞	∞	∞	4	∞	2	∞	3
i	∞	∞	∞	∞	∞	1	∞	3	∞

3. Wyjmij ze zbioru K krawędź o najmniejszej wadze, która łączy pewien wierzchołek w D z wierzchołkiem nienależącym do D . Dodaj tę krawędź do D wraz z wierzchołkiem, do którego prowadzi. Powtarzaj krok do momentu, gdy wszystkie wierzchołki należą do D .

Działanie algorytmu przedstawiamy krok po kroku na rys. 5.27. Liczby przy krawędziach oznaczają ich wagę. Startujemy z punktu w lewym dolnym rogu. Kolejne rysunki odpowiadają krokom algorytmu – do istniejącego drzewa doczepiamy kolejne krawędzie o możliwie najniższej wadze. Otrzymane w wyniku minimalne drzewo spinające ma wagę 15. Widzimy, że jest to rzeczywiście minimalna waga dla drzewa spinającego badanego grafu. Nieużyte zostały wszystkie trzy krawędzie z wagami 4 oraz jedna krawędź z wagą 3, a drzewo zawiera krawędzie



Rysunek 5.28: Kolejne kroki algorytmu Kruskala. Liczby oznaczają wagi krawędzi. Krawędzie wytłuszczone budują minimalne drzewo spinające

o wagach 1, 2 i 3. Można pokazać, że złożoność algorytmu wynosi $k + n \log n$, gdzie k jest liczbą krawędzi, a n liczbą wierzchołków.

Inny algorytm o tej samej funkcjonalności pochodzi od Kruskala¹⁴:

Alg. 5.5 (algorytm Kruskala znajdowania minimalnego drzewa spinającego). Niech G będzie grafem spójnym z wagami

1. Utwórz drzewo D składające się z wszystkich wierzchołków G .
2. Utwórz zbiór wszystkich krawędzi K .
3. Wyjmij ze zbioru K krawędź o najmniejszej wadze, dodaj do D , jeśli nie utworzy cyklu, w przeciwnym razie ją wyrzuć. Powtarzaj krok do momentu, gdy K jest pusty.

Jak widzimy, w rozważanym przykładzie w wyniku działania algorytmu powstało to samo końcowe minimalne drzewo spinające, co na rys. 5.27, natomiast etapy pośrednie są w obu przypadkach inne (por. 5.28 i 5.27). Złożoność algorytmu Kruskala wynosi $k \log k$, gdzie k jest liczbą krawędzi.

Podkreślimy jeszcze, że minimalne drzewo spinające jest drzewem o najkrótszej sumarycznej wadze, natomiast nie minimalizuje ono drogi między dowolnymi dwoma wierzchołkami. Na przykład na rys. 5.27 odległość od wierzchołka w lewym dolnym rogu do jego sąsiada po prawej stronie wynosi 4 wzdłuż łączącej te wierzchołki krawędzi, a wzdłuż minimalnego drzewa spinającego ta odległość wynosi 5, czyli jest dłuższa.

¹⁴ Joseph Bernard Kruskal (1929-), amerykański matematyk.

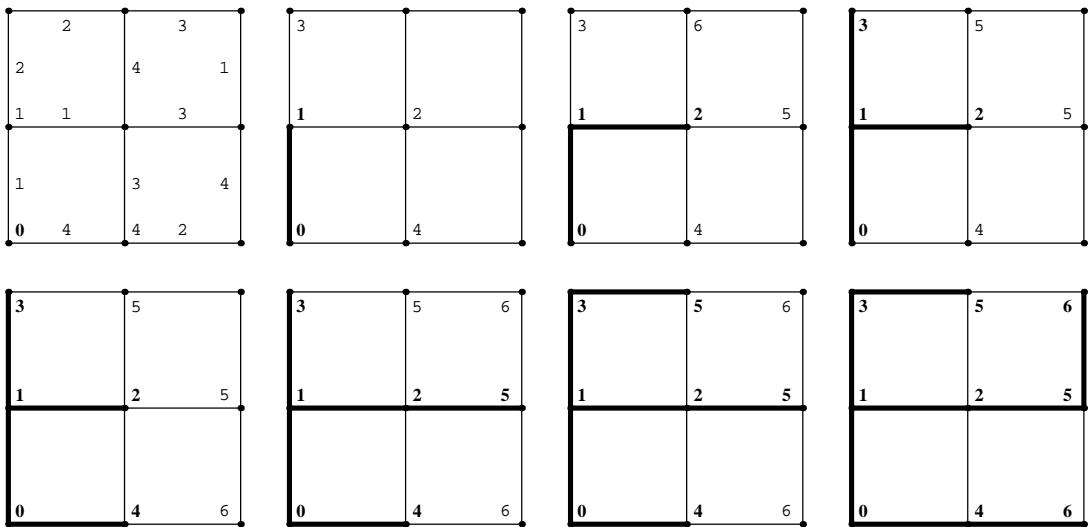
Problem znalezienia najkrótszej ścieżki między dwoma wierzchołkami grafu o *nieważnych* wagach rozwiązuje algorytm Dijkstry. Algorytm ten (i jego warianty) jest podstawowym narzędziem optymalizacji komunikacji internetowej (problem routingu) czy nawigacji samochodowej, więc warto go dokładnie poznać i zrozumieć. Algorytm wprowadza dodatkową charakterystykę wierzchołka grafu, zwyczajowo zwaną *etykietą*. Etykietą jest niczym innym, jak najkrótszą znalezioną po kolejnych krokach algorytmu odległośćą od wybranego wierzchołka startowego w_0 . Etykietą jest *tymczasowa*, poki może ulegać zmianie, i *stała*, gdy jest ostateczną najmniejszą odległością od w_0 .

Alg. 5.6 (algorytm Dijkstry znajdowania najkrótszej ścieżki (1956)).
Niech G będzie grafem spójnym z wagami nieujemnymi

1. Wybierz wierzchołek w_0 , dla którego chcemy znaleźć najkrótsze drogi do pozostałych wierzchołków. Oznacz ten wierzchołek tymczasową etykietą 0 (odległość wierzchołka od samego siebie jest 0). Wszystkim pozostałym wierzchołkom przydziel etykietę tymczasową ∞ (w tym momencie nie znamy jeszcze połączeń do pozostałych wierzchołków, więc są one „nieskończenie dalekie”). Utwórz początkowe drzewo D jako graf pusty złożony z wierzchołków grafu G .
2. Spośród wierzchołków o tymczasowych etykietach wybierz ten o najmniejszej etykiecie, oznaczony jako w . Oznacz jego etykietę jako etykietę stałą. Włącz krawędź, która prowadziła do tego wierzchołka przy nadawaniu mu ostatniej tymczasowej etykiety do drzewa D . Idź po każdej krawędzi wychodzącej z w do jej koncowego wierzchołka v o etykiecie tymczasowej T i oblicz liczbę L będącą sumą etykiety wierzchołka w i wagi krawędzi (w, v) . Jeśli ta liczba jest mniejsza od T , przydziel wierzchołkowi v nową etykietę tymczasową L .
3. Powtarzaj krok 2 do momentu, gdy wszystkie wierzchołki mają etykiety stałe. Po zakończeniu etykiety zawierają wagę najkrótszej drogi prowadzącej od danego wierzchołka do w_0 . Droga ta prowadzi wzduż powstałego drzewa spinającego D .

Można udowodnić, że algorytm ten jest poprawny, tj. zawsze prowadzi do rozwiązania. Jego złożoność obliczeniowa wynosi n^2 , gdzie n jest liczbą wierzchołków.

Prześledźmy algorytm Dijkstry na przykładzie z rys. 5.29. Pierwsza część rysunku ukazuje również wagi krawędzi. Początkowe drzewo D to graf pusty składający się z wierzchołków grafu. Powiedzmy, że interesują nas najmniejsze odległości od wierzchołka w lewym dolnym rogu, który oznaczamy w_0 i przypisujemy mu tymczasową etykietę 0. Pozostałym wierzchołkom przydzielamy tymczasową etykietę ∞ (nie zaznaczaną na rysunku). Przechodzimy do kroku 2. Bierzemy wierzchołek o najmniejszej etykiecie tymczasowej, jest to oczywiście w_0 oznaczony 0, bo na razie mamy tylko jeden oznaczony wierzchołek, oraz zmieniamy status etykiety na *stały*, co oznaczamy wytłuszczeniem czcionki. Następnie



Rysunek 5.29: Kolejne kroki algorytmu Dijkstry (należy czytać od lewej do prawej strony, najpierw górną, a potem dolny wiersz). Pierwszy rysunek pokazuje wagę krawędzi, które są jednakowe dla wszystkich części rysunku. Liczby przy wierzchołkach podają etykietę, a czcionka wytłuszczonego oznacza etykiety stałe. Etykiety nieskończone nie są zaznaczone

idziemy do wierzchołków połączonych z w_0 i nadajemy im etykiety będące wagami krawędzi łączących je z w_0 : 1 i 4. Jest to sytuacja z pierwszej części rys. 5.29. Następnie powtarzamy krok 2 algorytmu. Spośród wierzchołków o etykietach tymczasowych wybieramy jako w ten o najmniejszej etykiecie, czyli 1, zmieniamy status etykiety na stały (wytłuszczamy) oraz dodajemy krawędź, która doprowadziła do w z tworzonego drzewa spinającego D (wytłuszczenie). Obecna sytuacja pokazana jest na drugiej części rys. 5.29. Krawędzie z w prowadzą do dwóch wierzchołków o etykietach tymczasowych ∞ : w górę i w prawo. Dla każdego z tych wierzchołków obliczamy nowe etykiety L , będące sumą etykiety w (równie 1) oraz wagi krawędzi z w do wierzchołka. W wyniku dostajemy etykiety 2 i 3. Następnie kontynuujemy krok 2. I tak spośród trzech wierzchołków o etykietach tymczasowych 2, 3 i 4 (druga część rys. 5.29) wybieramy 2 jako w , wytłuszczamy etykietę i prowadzącą do w krawędź (trzecia część rysunku), po czym wyliczamy etykiety dla trzech krawędzi prowadzących w górę, w prawo i w dół z w do wierzchołków o etykietach tymczasowych. Dla wierzchołka poniżej nowa etykieta wynosi 5, więc jest większa od poprzedniej wynoszącej 4, w związku z tym nie ulega zmianie. Etykiety pozostałych wierzchołków połączonych z w były ∞ , więc przybierają nowe wartości 4 i 5. Następnie kontynuujemy krok 2 jeszcze pięć razy, patrz kolejne części rys. 5.29, aż do momentu, gdy wszystkie etykiety są stałe. Ich wartości ukazują odległość (wagę drogi prowadzącej do w_0). Wytłuszczone krawędzie zaznaczają drzewo spinające D .

Zauważmy, że jest to inne drzewo niż minimalne drzewo spinające z rys. 5.27 czy 5.28. Waga drzewa z rys. 5.29 wynosi 16, podczas gdy waga minimalnego drzewa spinającego jest równa 15. Minimalne drzewo spinające optymalizuje długość całego drzewa, podczas gdy algorytm Dijkstry optymalizuje odległości od wybranego wierzchołka.

Można pokazać, że złożoność obliczeniowa optymalnej implementacji algorytmu Dijkstry dla grafu o k krawędziach i n wierzchołkach wynosi $\mathcal{O}(k+n \log n)$.

Podobne zadanie do algorytmu Dijkstry wykonuje algorytm Bellmana¹⁵–Forda¹⁶, który jest ogólniejszy, ponieważ dopuszcza wagi ujemne. Można się bowiem łatwo przekonać, że algorytm Dijkstry załamuje się dla przypadku wag ujemnych. Rozważmy kontrprzykład będący modyfikacją sytuacji z rys. 5.29, gdzie wagę krawędzi między górnym lewym i górnym środkowym wierzchołkiem ustalamy na -6 zamiast 2 (zob. rys. 5.30). Do trzeciej części rysunku algorytm przebiega tak samo. Na czwartej części zamiast wagi 5 dla górnego środkowego wierzchołka pojawi się -3 . Jest oczywiste, że dotarcie do środkowego wierzchołka grafu, posiadającego etykietę *stałą* 2, jest lepsze z kierunku od góry, co dałoby mu wagę 1. Zatem przedwczesne było przydzielenie temu wierzchołkowi etykiety stałej. Algorytm Dijkstry załamuje się!

Zwrócić uwagę, że jeśli w grafie występuje krawędź o ujemnej wadze, to chodzenie po niej tam i z powrotem powoduje nieograniczony „zysk”. Aby temu zapobiec, niech krawędź od gh na rys. 5.30 będzie *skierowana*, co oznaczamy strzałką. Ogólnie, dla omawianych poniżej algorytmów, problem najkrótszej ścieżki ma sens tylko wtedy, gdy graf (skierowany) nie posiada pętli o ujemnej sumarycznej wadze.

Alg. 5.7 (algorytm Bellmana–Forda znajdowania najkrótszej ścieżki od danego wierzchołka). Niech G będzie grafem spójnym z wagami o n wierzchołkach.

1. Wybierz wierzchołek w_0 , dla którego chcemy znaleźć drogi o najmniejszej wadze prowadzące do pozostałych wierzchołków. Przydziel w_0 etykietę 0, a pozostałym wierzchołkom ∞ . Utwórz początkowe drzewo D jako graf pusty złożony z wierzchołków G .
2. Przeczesywanie.

Powtarzaj dla $i = 1, 2, \dots, n$: wybierz wierzchołek $w_i \neq w_0$.

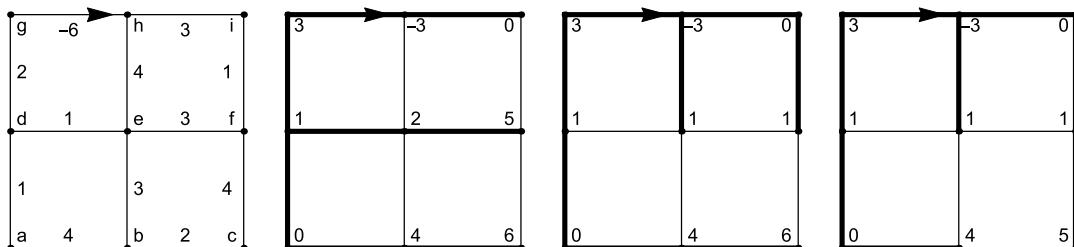
Powtarzaj dla j takich, że w_i i w_j są sąsiednie: jeśli etykieta w_i jest większa od liczby L , będącej sumą etykiety w_j i wagi krawędzi (w_i, w_j) , to przydziel w_i nową etykietę L . Usuń z D (jeśli istnieje) poprzednią krawędź prowadzącą do w_i , natomiast dodaj doń krawędź (w_i, w_j) .

¹⁵ Richard Ernest Bellman (1920-1984), amerykański matematyk.

¹⁶ Lester Randolph Ford, Jr. (1927–), amerykański matematyk.

3. Powtarzaj krok 2 aż do momentu, gdy etykiety nie ulegają dalszej zmianie. Po zakończeniu algorytmu etykiety zawierają wagę optymalnej drogi prowadzącej od danego wierzchołka do w_0 . Droga ta prowadzi wzduż powstały drzewa spinającego D .

Prześledźmy działanie algorytmu Bellmana–Forda na przykładzie z rys. 5.30. Czytelnik powinien samodzielnie wykonać odpowiednie rysunki. W rozważanym grafie jedna krawędź, (g, h) , ma wagę ujemną. Wybieramy $w_0 = a$ i przydzielamy mu etykietę 0, pozostałe wierzchołki mają etykietę ∞ . Drzewo D inicjalizujemy jako graf pusty składający się z dziewięciu wierzchołków a, b, \dots, i . Rozpoczynamy „przeczesywanie” grafu. Rozpoczynamy pierwszą pętlę po wierzchołkach. Bierzemy wierzchołek b , rozpoczynamy drugą pętlę po wierzchołkach w połączonych z b . Dla $w = a$ liczba L , będąca sumą etykiety a wynoszącej 0 oraz wagi krawędzi (a, b) wynosi 4, co jest mniejsze od dotychczasowej etykiety b wynoszącej ∞ , więc przydzielamy b etykietę $L = 4$. Dodajemy krawędź (a, b) do D . Pozostałe wierzchołki połączone z b , mianowicie c i e , mają etykietę ∞ i nie zmniejszają etykiety b . Kończymy więc pętlę drugą i wracamy do pierwszej. Bierzymy $w = c$. Rozpoczynamy pętle drugą po wierzchołkach połączonych z w . Ponieważ c jest połączony z b , uzyskuje etykietę $4 + 2 = 6$. Dodajemy krawędź (b, c) do D . Wracamy do pętli pierwszej i kontynuujemy proces dla pozostałych wierzchołków d, \dots, i . Po zakończeniu pierwszego przeczesania sytuacja wygląda jak na drugiej części rys. 5.30. Rozpoczynamy teraz drugie przeczesanie. Nic się nie zmienia, aż dochodzimy w pierwszej pętli do $w = e$, posiadającego etykietę 2. Zauważamy, że dojście do tego wierzchołka od góry, przez wierzchołek h , jest lepsze i daje etykietę $L = -3 + 4 = 1$, mniejszą od dotychczasowej. W związku z tym nadajemy wierzchołkowi e etykietę 1, usuwamy dotychczasową krawędź, która prowadziła do e , czyli (d, e) , z drzewa D , a na jej miejsce dodajemy do D krawędź (d, h) . Podobna sytuacja ma miejsce dla wierzchołka f . Po zakończeniu drugiego przeczesania sytuacja wygląda jak na trzeciej części rys. 5.30. W trzecim przeczesaniu zmianie ulega status wierzchołka c . Dalsze przeczesywanie nie prowadzi już do żadnych zmian i ostateczna sytuacja jest taka, jak na



Rysunek 5.30: Graf z wagą ujemną (pierwszy rysunek) oraz sytuacja po kolejnych przeczesaniach z algorytmu Bellmana–Forda. Liczby przy wierzchołkach oznaczają etykiety, tj. dotychczas znalezione drogi o najmniejszej wadze od lewego dolnego wierzchołka, a wyłuszczone krawędzie odpowiadają budowanemu minimalnemu drzewu spinającemu

Tabela 5.4: Tablica minimalnych odległości między wierzchołkami uzyskana za pomocą algorytmu Floyda–Warshalla dla grafu z wagami z tabeli 5.3

	a	b	c	d	e	f	g	h	i
a	2	4	6	1	2	5	3	5	6
b	4	4	2	4	3	6	6	7	7
c	6	2	4	6	5	4	8	8	5
d	1	4	6	2	1	4	2	4	5
e	2	3	5	1	2	3	3	4	4
f	5	6	4	4	3	2	6	4	1
g	3	6	8	2	3	6	4	2	5
h	5	7	8	4	4	4	2	4	3
i	6	7	5	5	4	1	5	3	2

czwartej części rys. 5.30. Widzimy, że drogi z a do odleglejszych wierzchołków „lubią” przechodzić po krawędzi (g, h), gdyż zyskują wtedy na ujemnej wadze tej krawędzi.

Złożoność obliczeniowa algorytmu Bellmana–Forda wynosi $\mathcal{O}(kn)$, gdzie k jest liczbą krawędzi, a n liczbą wierzchołków grafu, jest więc większa od złożoności algorytmu Dijkstry. Dla grafów o dodatnich wagach należy wobec tego używać algorytmu Dijkstry.

Modyfikacją algorytmu Dijkstry dla grafów o nieujemnych wagach jest algorytm Floyda¹⁷–Warshalla¹⁸, znajdujący za jednym zamachem najkrótsze odległości między wszystkimi parami wierzchołków w grafie z wagami. Zapiszemy go nieco bardziej skrótnie niż dla poprzednich algorytmów i tylko dla przypadku podającego minimalne odległości, a nie ścieżki między parami wierzchołków.

Alg. 5.8 (algorytm Floyda–Warshalla znajdowania najkrótszych odległości dla wszystkich par wierzchołków grafu o nieujemnych wagach). Niech G będzie grafem spójnym z tablicą nieujemnych wag w . Wykonaj potrójną pętlę po wskaźnikach k, i, j : dla $k = 1, \dots, n, i = 1, \dots, n, j = 1, \dots, n$ jeżeli $w_{ik} + w_{kj} < w_{ij}$, zastąp w_{ij} przez $w_{ik} + w_{kj}$. Po zakończeniu tablica w zawiera minimalne wagi dróg między parami wierzchołków.

Trzy zagnieżdżone pętle prowadzą do złożoności obliczeniowej $\mathcal{O}(n^3)$. Jako przykład rozważmy tablicę wag 5.3. Algorytm Floyda–Warshalla wykonany dla tego przypadku daje w wyniku tablicę minimalnych odległości między wierzchołkami ukazaną w tabeli 5.4. Wyrazy na przekątnych odpowiadają wagom minimalnym dróg wychodzących z danego wierzchołka i doń powracających. Jest to więc podwojona najmniejsza waga krawędzi wychodzącej z tego wierzchołka.

¹⁷ Robert Floyd (1936-2001), amerykański informatyk.

¹⁸ Stephen Warshall (1935-2006), amerykański informatyk.

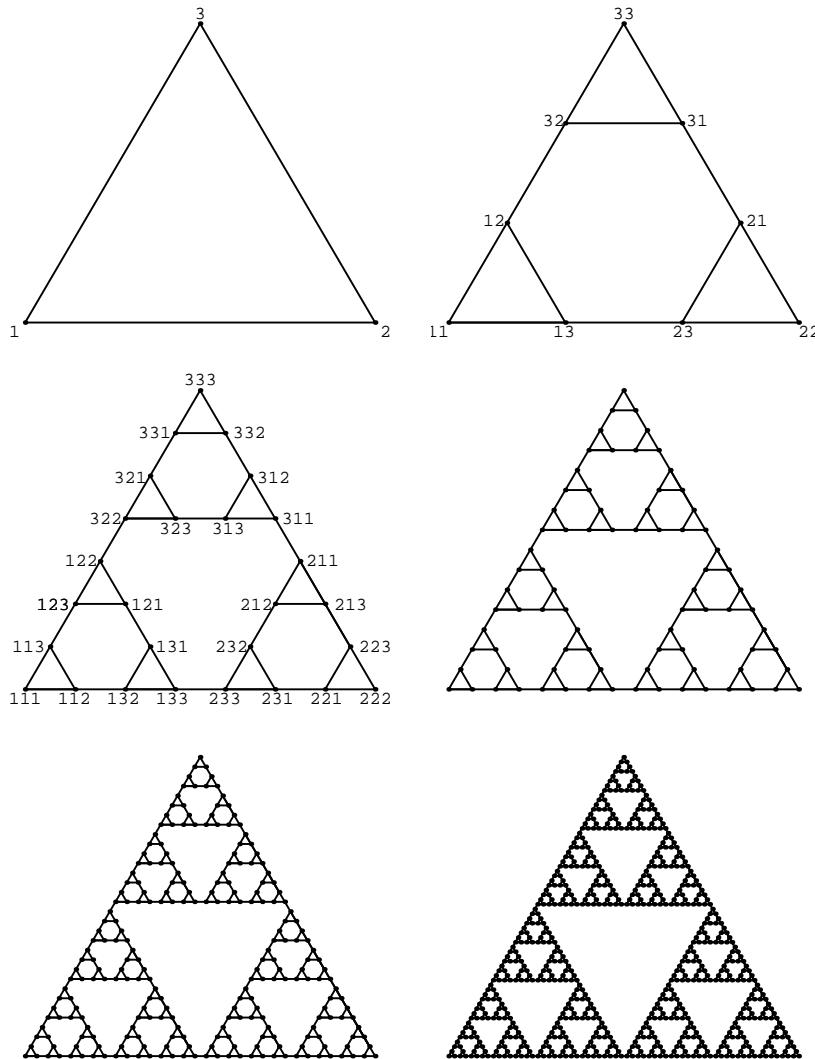
5.9 Jeszcze o wieżach Hanoi

Umiejętność posługiwania się teorią grafów pozwala na podanie alternatywnego dowodu Tw. 1.1, a mianowicie, że algorytm rozwiązania problemu Wież Hanoi 1.1 jest w istocie najlepszym możliwym algorytmem. Jak przetłumaczyć problem Wież Hanoi na język grafów? Otóż wierzchołki identyfikujemy ze stanami, czyli układami krążków na prętach. Niech symbol $ab\dots c$ oznacza największy krążek na pręcie a , drugi co do wielkości na pręcie b, \dots , najmniejszy na c , np. 212 jest konfiguracją, gdzie średni krążek spoczywa na pręcie 1 , duży na pręcie na 2 , a na nim najmniejszy. Zauważmy, że reguła układania krążków z mniejszym wyżej unika niejednoznaczności, tzn. jeśli kilka krążków nanizanych jest na ten sam pręt i dana cyfra powtarza się w symbolu stanu, to rozumiemy, że kolejność ułożenia jest zgodna z regułą. Krawędzie grafu odpowiadają zgodnymi z regułami gry przejściami między stanami.

Zacznijmy od trywialnego przypadku jednego krążka, $n = 1$. Możliwe stany to 1 , 2 i 3 , mamy więc trzy wierzchołki. Wszystkie przejścia są dozwolone, odpowiada to po prostu przekładaniu jedynego krążka z pręta na pręt, a więc wszystkie wierzchołki są połączone krawędziami, patrz pierwsza część rys. 5.31. Dla dwóch krążków odpowiednia sytuacja pokazana jest na drugiej części rysunku. Mamy $3^n = 9$ stanów: $11, 12, 13, 21, 22, 23, 31, 32$ i 33 . Zauważmy, że nie wszystkie przejścia są dozwolone, tylko te, gdzie przekładamy jeden krążek. Na trzeciej części rysunku mamy przypadek trzech krążków, $n = 3$, dający $3^n = 27$ stanów. Zauważamy tu prawidłowość. Graf dla n powstaje z połączenia trzech grafów dla $n - 1$. Na przykład w lewej dolnej części grafu dla $n = 3$ rozpoznajemy, po „odcięciu” pierwszej cyfry w symbolu wierzchołka, czyli usunięciu największego krążka, graf dla $n = 2$. Mówiąc ściślej, układ graficzny lewej dolnej części grafu dla $n = 3$ powstaje z grafu dla $n = 2$ po odbiciu go na płaszczyźnie wzduż osi $y = x$. Dla $n \geq 4$ konstrukcja grafu przebiega analogicznie, co pokazano na kolejnych częściach rys. 5.31. W fascynujący sposób powstaje więc z Wież Hanoi trójkąt Sierpińskiego!¹⁹

Co nam daje ta analiza? Otóż widzimy natychmiast, że najkrótsza droga z wierzchołka $11\dots 1$, czyli „wszystkie krążki na pręcie 1”, do stanu $22\dots 2$, „wszystkie krążki na pręcie 2”, przebiega wzduż dolnej krawędzi grafu, bez żadnych „skoków w bok”, co w oczywisty sposób wydłużłyby drogę. Najpierw wykonujemy przejście podstawy lewego dolnego trójkąta, co sprowadza się do przeniesienia górnych $n - 1$ krążków z pręta 1 na pręt 3, potem przenosimy dolny krążek z pręta 1 na pręt 2 – jest to krawędź na środku podstawy całego trójkąta, łącząca lewą dolną i prawą dolną część trójkąta, wreszcie przekładamy $n - 1$ krążków z pręta 2 na pręt 3, idąc wzduż podstawy prawego dolnego trójkąta. Jest to właśnie nasz

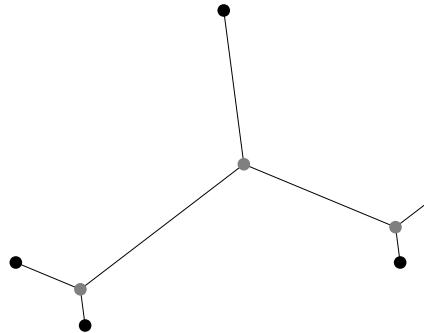
¹⁹ Wacław Sierpiński (1882-1969), polski matematyk, jeden z twórców warszawskiej szkoły matematycznej, znacznie przyczynił się do rozwoju teorii mnogości, topologii, teorii liczb i analizy matematycznej.



Rysunek 5.31: Grafy dla problemu Wież Hanoi o kolejnej liczbie krążków $n = 1, 2, \dots, 6$. Etykiety wierzchołków oznaczają konfigurację krążków na prętach, np. 122 oznacza największy krążek na przecie 1, a średni i najmniejszy na przecie 2

algorytm 1.1! Widzimy też, że liczba kroków całej operacji przeniesienia wynosi $2^n - 1$.

Liczba wierzchołków „grafa Hanoi” z rys. 5.31 spełnia rekurencję jednorodną $w_n = 3w_{n-1}$, $w_1 = 3$, co daje natychmiast $w_n = 3^n$. Jest to oczywiste, bo każdy z n krążków może spoczywać na jednym z trzech prętów. Natomiast dla liczby krawędzi mamy rekurencję niejednorodną $k_n = 3k_{n-1} + 3$, $k_1 = 3$. Stosując metodę z rozdz. 1.6, znajdujemy $k_n = \frac{2}{3}(3^n - 1)$. Gdybyśmy nie znali optymalnego algorytmu rozwiązania problemu Wież Hanoi, musielibyśmy poszukiwać najkrótszej



Rysunek 5.32: Przykładowe euklidesowe drzewo Steinera. Ciemne punkty oznaczają oryginalne wierzchołki grafu (studzienki naftowe), a jaśniejsze punkty wierzchołki dodane (trójkąty)

ścieżki między skrajnymi wierzchołkami grafu Hanoi. W tym celu można zastosować np. algorytm 5.6, ale ponieważ złożoność tego algorytmu rośnie z kwadratem liczby wierzchołków, czyli w naszym przypadku jak $\Theta(9^n)$, procedura jest dużo wolniejsza niż złożoność $\Theta(2^n)$ optymalnego algorytmu Hanoi 1.1. Widzimy tu więc namacalnie praktyczną rolę posiadania algorytmu *optimalnego* dla danego problemu. Nie zawsze jednak jesteśmy w takiej komfortowej sytuacji.

Uogólnienie problemu Wież Hanoi na przypadek czterech pretów (tzw. łamigłówka Reve'a, zob. zad. 5.13) prowadzi do grafu w naturalny sposób reprezentowalnego w trzech wymiarach przestrzennych. Okazuje się, że nie jest to tzw. piramida Sierpińskiego (uogólnienie trójkąta Sierpińskiego na trzy wymiary), ale dużo bardziej skomplikowany obiekt. Graf zawiera bowiem znacznie więcej połączeń między stanami niż piramida Sierpińskiego i ten fakt czyni jego analizę dramatycznie bardziej skomplikowanym zadaniem. Dla dużych n problem znalezienia najkrótszej ścieżki z narożnika do narożnika dla tego grafu jest trudny. Możemy go oczywiście rozwiązać, stosując np. algorytm Dijkstry, ale dla dużego n jest to bardzo czasochłonne. Istnieje nieudowodniona hipoteza o optymalnym algorytmie dla łamigłówki Reve'a, sprawdzona komputerowo do $n \sim 30$ [46].

5.10 Drzewo Steinera

Rozważmy następujący problem nafciarza z Teksasu: n studzienek pompujących ropę naftową należy połączyć rurami w taki sposób, aby ich sumaryczna długość była jak najmniejsza. Rury możemy prowadzić dowolnie, co więcej, możemy je łączyć/rozgałęziać trójkątkami. Koszt jednostki długości wszystkich rur jest jednakowy. Zagadnienie jest istotnie różne od opisanego wcześniej problemu znajdowania minimalnego drzewa spinającego, ponieważ łączenie rur powoduje powstawanie dodatkowych wierzchołków, których położenie jest optymalizowane. Ilustracja przedstawiona jest na rys. 5.32. Ciemne punkty oznaczają oryginalne wierzchołki grafu, odpowiadające położeniom studzienek naftowych, a jaśniejsze punkty oznaczają wierzchołki dodane, czyli połączenia rur za pomocą trójkątów.

Zagadnienie nosi nazwę problemu drzewa Steinera²⁰. Sformułowanie matematyczne jest następujące:

Def. 5.34 (euklidesowe drzewo Steinera). *Niech waga krawędzi będzie równa odległości euklidesowej wierzchołków. Mamy n oryginalnych wierzchołków o ustalonych położeniach na płaszczyźnie. W jaki sposób rozmieścić pomocnicze wierzchołki, zwane wierzchołkami Steinera, i jak dobrać krawędzie, aby znaleźć minimalne drzewo spinające grafu zawierającego wierzchołki ustalone i pomocnicze.*

Rozważa się też uogólnienia problemu na inne metryki oraz na przypadek trójwymiarowy. Problem znalezienia minimalnego drzewa Steiner jest trudny²¹ (patrz rozdz. 6.6). Ze względu na jego duże znaczenie praktyczne w zagadnieniach optymalizacyjnych jest przedmiotem intensywnych badań algorytmicznych.

Rozważmy na początek najprostszą sytuację, gdy mamy tylko trzy oryginalne wierzchołki grafu. Problem sprowadza się więc do znalezienia punktu na płaszczyźnie, dla którego suma odległości od trzech wierzchołków trójkąta jest minimalna. To geometryczne zagadnienie zostało postawione przez Fermata i rozwiązane w 1640 r. przez Torricelliego²² oraz niezależnie przez innych współczesnych mu matematyków. Szukany punkt Torricelliego jest wierzchołkiem Steinera dla przypadku grafu o trzech oryginalnych wierzchołkach.

Zamiast przytaczać skądinąd eleganckie geometryczne rozważania siedemnastowiecznych matematyków, w celu rozwiązania problemu Fermata posłużymy się prawami mechaniki i skonstruujemy komputer analogowy, który w zadziwiająco prosty sposób znajdzie optymalne rozwiązanie. Komputer analogowy to urządzenie, które wykonuje obliczenia nie poprzez działania na liczbach, ale dzięki zastosowaniu w sprytny sposób elementów mechanicznych, elektronicznych itp. Oto nasz komputer rozwiązujący problem Fermata (podobną maszynkę zaproponowali Pick²³ i Pólya²⁴, zob. [47]):

Alg. 5.9 (komputer analogowy dla problemu Fermata).

1. Wywierć wiertarką w stole trzy otwory w miejscach odpowiadającym wierzchołkom trójkąta.
2. Przewlecz przez otwory trzy kawałki nici i zwiąż je w jednym punkcie nad stołem (wygodniej jest przewlec jeden kawałek nici przez otwory 1 i 2, i dowieźać doń kawałek przechodzący przez otwór 3).
3. Pod stołem przywiąż do każdej nici ciężarek o takiej samej wadze tak, aby mógł luźno zwisać.

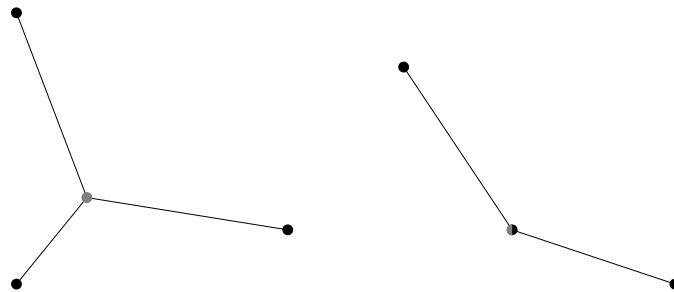
²⁰ Jakob Steiner (1796–1863), szwajcarski matematyk.

²¹ Należy do klasy złożoności NP.

²² Evangelista Torricelli (1608–1647), włoski fizyk i matematyk, skonstruował barometr rtęciowy.

²³ Georg Alexander Pick (1859-1942), austriacki matematyk.

²⁴ George Pólya (1887-1985), węgierski matematyk, istotnie przyczynił się do rozwoju teorii liczb, kombinatoryki i rachunku prawdopodobieństwa.



Rysunek 5.33: Możliwe sytuacje dla rozwiązywania problemu Fermata: punkt Torricelliego (jasny kolor) jest w innym położeniu niż wierzchołki trójkąta (lewa strona) lub pokrywa się z położeniem jednego z wierzchołków trójkąta

4. Układ ustawi się w położeniu równowagi. Punkt związania nici jest punktem Torricelliego!

Dlaczego to działa? Z mechaniki wiemy, że układ dąży do stanu równowagi, w którym *energia potencjalna* jest jak najmniejsza. Dla jednego ciała wielkość ta równa się iloczynowi jego masy i wysokości ponad ustalonym poziomem (np. podłogi). Dla kilku ciał energia potencjalna się dodaje. W naszym przypadku wynosi więc

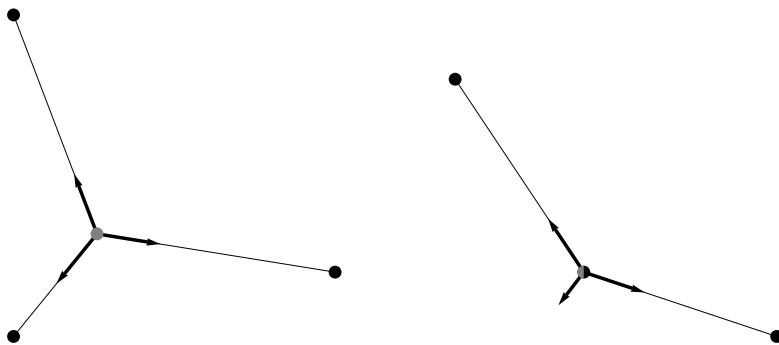
$$mh_1 + mh_2 + mh_3 = m(h_1 + h_2 + h_3),$$

gdzie m jest masą każdego ciężarka, a h_i odległością i -tego ciężarka od podłogi. Tak więc suma odległości od podłogi, $h_1 + h_2 + h_3$, osiąga w równowadze minimalną wartość. Wówczas sumaryczna długość nici pod stołem jest jak największa, a tym samym na stole jak najmniejsza. A o to właśnie chodzi w znajdowaniu drzewa Steinera!

Przy wykonywaniu algorytmu 5.9, w zależności od tego jak ulokowane są wierzchołki trójkąta, możliwe są dwie sytuacje: położenie punktu Torricelliego będzie inne niż położenia wierzchołków trójkąta lub będzie się pokrywać z niektórymi z nich (rys. 5.33). Posłużymy się teraz dalej prawami mechaniki i dorysujmy siły działające na punkt związania nici. Ponieważ punkt ten jest w spoczynku, siły nań działające muszą się równoważyć. Wartość każdej z tych sił jest taka sama, gdyż równa się ciężarowi uwiązanego do nici ciężarka, a wszystkie ciężarki są jednakowe. Aby doszło do zrównoważenia trzech sił o jednakowej wartości, kąt między nimi musi wynosić 120° ²⁵. Warunek ten nosi nazwę warunku kąta. Zauważmy też, że aby mogła zajść sytuacja z lewej strony rys. 5.34, wszystkie kąty rozważanego trójkąta muszą być mniejsze od 120° . Jeśli tak nie jest (prawa strona rys. 5.34), wówczas punkt Torricelliego pokrywa się z tym wierzchołkiem trójkąta, w którym kąt jest większy od 120° .

Wyprowadziliśmy więc następujące twierdzenie dla problemu Fermata:

²⁵ Dla nieobeznanych ze szkolną fizyką, siły dodajemy jak wektory. Aby suma trzech wektorów o równej długości wyniosła zero, kąt między nimi musi wynosić 120° .



Rysunek 5.34: To samo, co na rys. 5.33, z dorysowanymi siłami. W sytuacji z lewej strony kąty między siłami wynoszą 120° . Na rysunku po prawej stronie mniejsza wartość trzeciej siły wynika z oddziaływania nici z otworem w stole

Tw. 5.12 (warunek kąta dla problemu Fermata). *Jeśli wszystkie kąty trójkąta są mniejsze niż 120° , wówczas punkt Torricelliego jest w takim położeniu, że odcinki łączące go z wierzchołkami trójkąta tworzą kąty 120° . Jeśli jeden z kątów trójkąta jest większy niż 120° , wtedy punkt Torricelliego pokrywa się z wierzchołkiem tego kąta.*

Teraz przejdźmy do ogólnego problemu Steinera o n oryginalnych wierzchołkach. Dodatkową komplikacją w stosunku do przeanalizowanego przypadku z $n = 3$ jest możliwość różnych *topologii* połączeń. Zilustrowane jest to na rys. 5.35. Dwa górne rysunki pokazują optymalne rozwiązania dla dwóch topologii, gdzie krawędzie się nie krzyżują. Liczby na rysunkach wskazują długość uzyskanego drzewa Steinera. W dolnej części rysunku ukazana jest sytuacja topologii, gdzie krawędzie się krzyżują. W tym przypadku optymalne rozwiązanie (proces dochodzenia do niego jest schematycznie zaznaczony strzałkami od lewej do prawej strony) uzyskane jest dla przekrywających się wierzchołków Steinera. Natomiast najlepszym rozwiązaniem dla rozważanego przykładu jest topologia z górnej prawej części rysunku. Tak więc mamy tu de facto dwa problemy: znaleźć wszystkie topologie, oraz dla każdej topologii znaleźć optymalne położenia wierzchołków Steinera.

Podamy teraz użyteczne fakty dotyczące problemu Steinera o n oryginalnych wierzchołkach. Dla prostoty zliczania traktujemy pokrywające się wierzchołki Steinera jako oddzielne wierzchołki połączone krawędzią o długości 0. Zauważmy też, że wierzchołki oryginalne są liśćmi drzewa Steinera oraz że wierzchołki Steinera mają stopień 3.

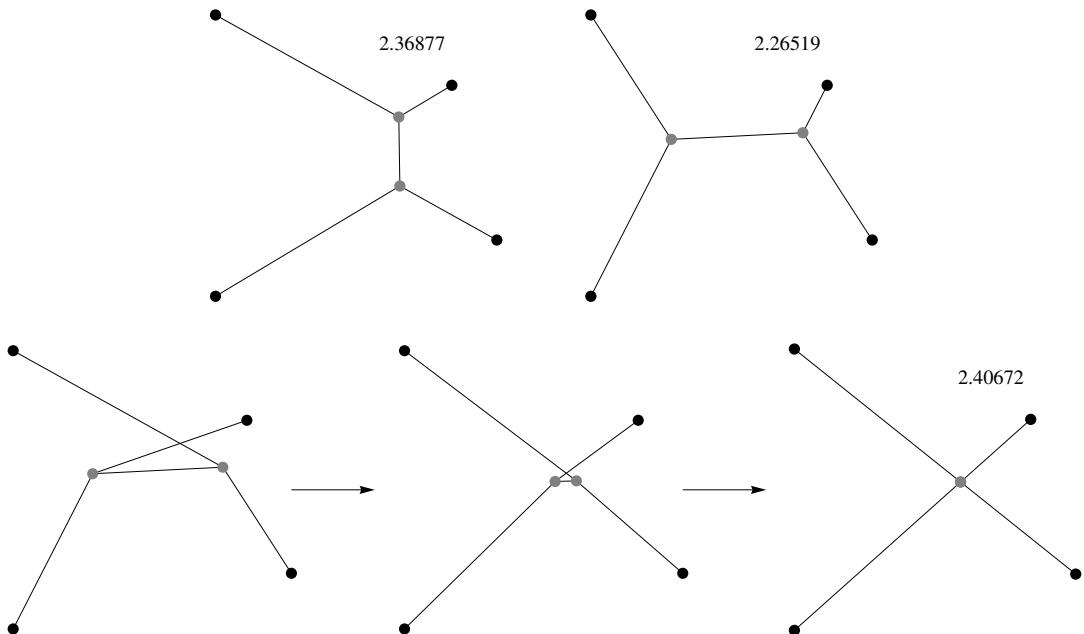
Tw. 5.13. *Drzewo Steinera o n oryginalnych wierzchołkach ma $n - 2$ wierzchołków Steinera, $2n - 3$ krawędzi oraz $t_n = (2n - 5)!!$ możliwych topologii.*

Dowód: Dowód jest indukcyjny. Teza zachodzi dla $n = 3$, co widzimy z lewej części rys. 5.33. Następnie zakładamy, że twierdzenie zachodzi dla $n - 1$ oryginalnych wierzchołków. W kroku indukcyjnym dodajemy parę wierzchołek oryginalny – wierzchołek Steinera, połączone krawędzią. W oczywisty sposób liczba wierzchołków Steinera wzrasta o 1, a liczba krawędzi o 2, bo jedna krawędź została przecięta. To dowodzi dwóch pierwszych części twierdzenia, pozostała wzór na liczbę topologii. Nowy wierzchołek Steinera możemy umieścić na dowolnej z $2n - 5$ krawędzi (zob. rys. 5.36 i 5.37). Tak więc $t_n = (2n - 5)t_{n-1}$ i liczba topologii wynosi $(2n - 5) \cdot (2n - 7) \dots 3 \cdot 1$. \square

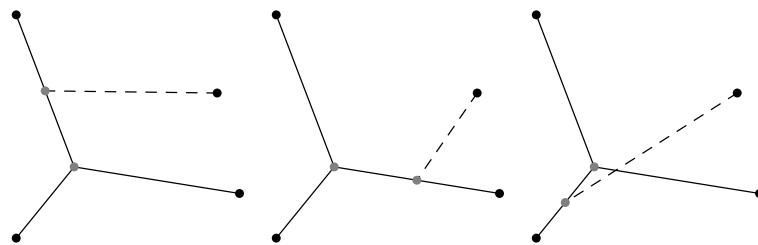
Możemy teraz przystąpić do konstrukcji komputera analogowego dla ogólnego problemu drzewa Steinera. W porównaniu z algorytmem 5.9 dla $n = 3$ mamy tu dwie różnice: musimy rozważyć różne topologie, ponadto nie możemy wiązać nici na stałe, bo uniemożliwiłoby to zmianę odległości między punktami Steinera. Postępowanie jest następujące:

Alg. 5.10 (komputer analogowy dla problemu Steinera).

1. Przygotuj n jednakowych ciężarków.
2. Wywierć wiertarką w stole otwory w miejscach odpowiadającym oryginalnym wierzchołkom grafu.
3. Przewlecz przez dowolne dwa otwory nić, pod stołem przywiąż do obu końców ciężarki.

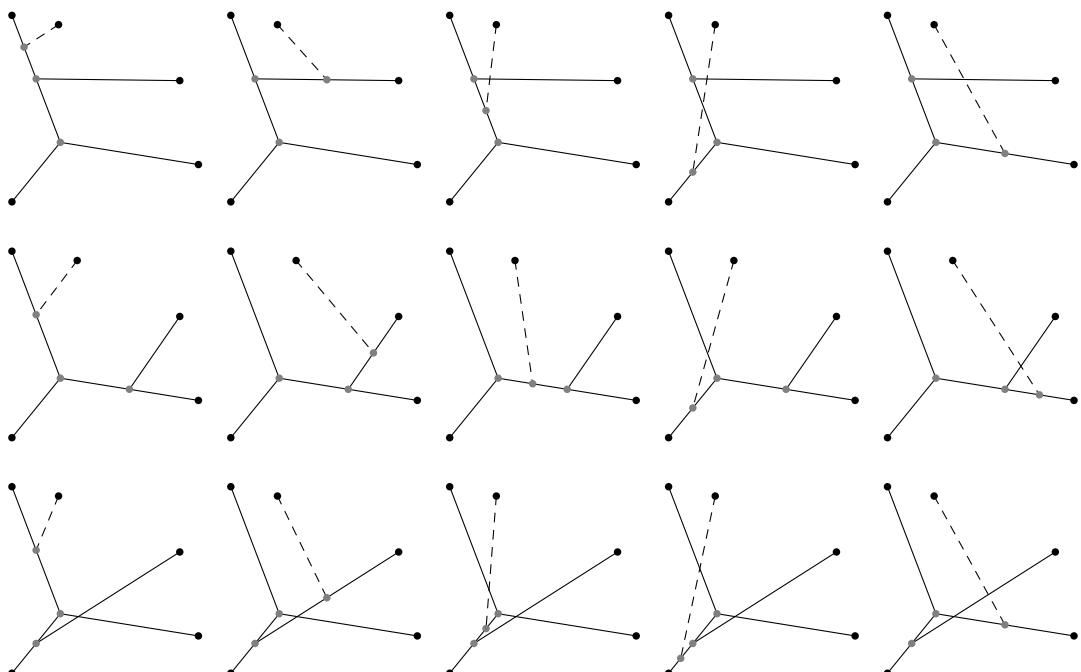


Rysunek 5.35: Różne topologie połączeń dla problemu Steinera o czterech oryginalnych wierzchołkach. Liczby ukazują otrzymaną długość drzewa Steinera dla danej topologii

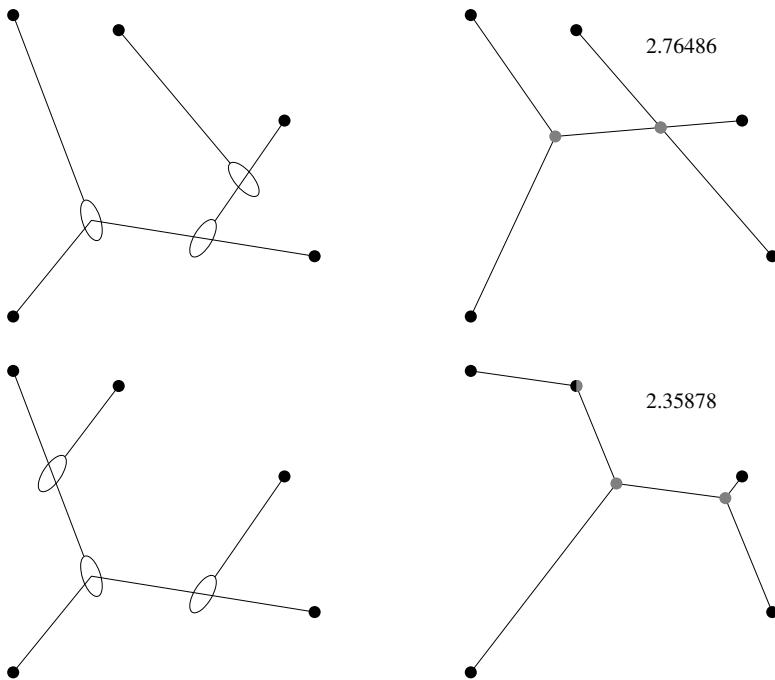


Rysunek 5.36: Ilustracja kroku indukcyjnego przy zliczaniu topologii dla $n = 4$. Nowy wierzchołek Steinera można umieścić na każdej krawędzi grafu o $n = 3$, tj. na $(2n - 5)!! = 3$ sposoby

4. Do nici na stole przywiąż za pomocą pętelki kolejną nić (zob. lewa strona rys. 5.38) w taki sposób, aby mogła się luźno przesuwać. Jej koniec wpuść do niezajętego otworu i pod stołem przywiąż ciężarek (zamiast pętelek można też użyć spinaczy jako haczyków).
5. Powtórz poprzedni krok $n - 3$ razy, tak aby przez każdy otwór przechodziła jedna nić.
6. Dzięki działaniu sił grawitacji na ciężarki układ ustawi się w położeniu równowagi. Zanotuj sumaryczną długość drzewa Steinera L .



Rysunek 5.37: To samo, co na rys. 5.36 dla $n = 5$, dającego $5!! = 15$ topologii. Grafy w każdym wierszu utworzone są poprzez dodanie wierzchołka Steinera na krawędziach jednego z grafów z rys. 5.36



Rysunek 5.38: Sposób wiązania nici dla wybranej topologii (lewa strona) oraz wynik rachunku wykonanego przez komputer analogowy (prawa strona). Liczba na rysunku podaje sumaryczną długość drzewa. Góra część: topologia z drugiego wiersza i drugiej kolumny rys. 5.36, dolna część: topologia z drugiego wiersza i pierwszej kolumny rys. 5.36, dająca rozwiązanie optymalne

7. Powtórz kroki 3-6 dla każdej z $(2n - 5)!!$ topologii (to jest uciążliwa część tego algorytmu!).
8. Przypadek z najmniejszą wartością L jest optymalnym rozwiązaniem problemu Steinera. Punkty połączenia nici ustalone są w wierzchołkach Steinera.

Przykład działania algorytmu dla $n = 5$ przedstawiony jest na rys. 5.38. W górnej części rysunku rozważana jest topologia z drugiego wiersza i drugiej kolumny rys. 5.36. Po lewej stronie pokazany jest sposób powiązania nici z pętelkami umożliwiającymi zmianę wzajemnych odległości między punktami Steiner. Po uruchomieniu komputera układ ustawi się w położeniu równowagi, jak w prawej górnej części rysunku. W rozważanej topologii dwa wierzchołki Steinera przekrywają się. Liczba na rysunku wskazuje sumaryczną długość drzewa Steinera. Przypadek topologii z górnej części rysunku nie jest optymalny. Dla topologii z drugiego wiersza i pierwszej kolumny rys. 5.36 ta długość jest krótsza, co указанo jest w dolnej części rys. 5.38. Sprawdziwszy 15 topologii, przekonujemy się, że to rozwiązanie jest istotnie optymalne. Można ogólnie pokazać, że jeśli dwa punkty Steinera pokrywają się, to rozwiązanie z tą topologią nie jest optymalne.

Ze względu na konieczność rozważania wielu topologii, nasz komputer analogowy nie jest zbyt praktyczny w zastosowaniu dla większych wartości n niż, powiedzmy, 6. Jest jednak bardzo użyteczny dla przeprowadzenia ogólnych rozważań. Na rys. 5.38 widzimy, że nici wokół wierzchołków Steinera nieprzekrywających się z innymi wierzchołkami tworzą kąty 120° . Wynika to z zasad mechaniki, przy czym dyskusja jest analogiczna jak w przypadku z rys. 5.34. Mamy więc następujące uogólnienie Tw. 5.12:

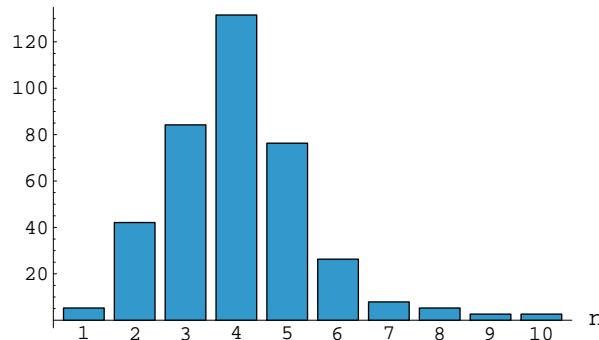
Tw. 5.14 (warunek kąta dla problemu Steinera). *W drzewie Steinera krawędzie wychodzące z wierzchołków Steinera nieprzekrywających się z innymi wierzchołkami tworzą kąty 120° . Jeśli punkt Steinera pokrywa się z wierzchołkiem oryginalnym, to kąt między wychodzącymi zeń krawędziami jest większy od 120° .*

Jako ciekawostkę wspomnijmy inny komputer analogowy stosowany do rozwiązywania problemu Steinera, który używa baniek mydlanych [48]! Wykorzystane jest tu zjawisko napięcia powierzchniowego. Urządzenie składa się z dwóch identycznych równoległych przeźroczystych płyt (np. z pleksi), w których w miejscach odpowiadających oryginalnym wierzchołkom grafu przewiercone są otwory, a w nich umocowane są (prostopadle do płyt) pręty. Całość urządzenia zanurza się w roztworze mydła. Po wyciągnięciu między prętami naciągnięte są błony mydlane. Miejsca styku błon z płaszczyzną płyty układają się w krawędzi grafu. Widzimy, że tworzą się również wierzchołki Steinera, odpowiadające połączeniom błon mydlanych. Napięcie powierzchniowe działa analogicznie do nici z ciężarkami z opisanego wyżej mechanicznego komputera analogowego. Dąży ono do zminimalizowania całkowitej powierzchni błony mydlanej, co przy ustalonej odległości między płytami jest równoważne minimalizacji długości powstałego drzewa. Podobnie jak dla nici w wersji mechanicznej, równowaga sił prowadzi do połączenia błon pod kątem 120° w izolowanych wierzchołkach Steinera oraz do kątów większych niż 120° , jeśli wierzchołek pokrywa się z prętem. Komputer mydlany wybiera topologię połączeń przy każdej próbie na chybień trafił, więc nie sprawujemy nad tym kontroli, ale przynajmniej nie musimy żmudnie wiązać nici jak w komputerze mechanicznym!

5.11 „Mały świat”

Koncepcje teorii grafów można z powodzeniem stosować do badania sieci społecznych [49]. W roku 1967 Stanley Milgram²⁶ przeprowadził następujący eksperyment: spośród mieszkańców miast Boston, Wichita i Omaha zarekrutowano uczestników będących osobami „startowymi” i „końcowymi” doświadczenia. Do osób startowych wysłano pakiety informacyjne, wyjaśniające celowość projektu,

²⁶ Stanley Milgram (1933–1984), amerykański psycholog i socjolog, znany z doświadczeń dotyczących „posłuszeństwa autorytetowi” oraz „Małego świata”.



Rysunek 5.39: Wyniki niedawnego doświadczenia „Małego świata” z pracy [50], gdzie uczestnicy kontaktowali się za pomocą poczty elektronicznej. Na osi poziomej liczba kroków, na pionowej liczba wiadomości, które osiągnęły cel. Najwięcej wiadomości dotarło już po $n = 4$ krokach, a dla 95% wiadomości wystarczyło 6 lub mniej kroków

a także niezbędne podstawowe informacje personalne o osobie końcowej, do której osoba startowa miała dotrzeć za pomocą łańcucha korespondencji. Zadanie polegało na wysłaniu załączonej karty pocztowej do dobrze znanej osoby (zdefiniowanej, jako osoba, z którą jest się „na ty”²⁷), wybranej zgodnie z intuicją, że zbliży nas do osoby-celu. Kolejne osoby w łańcuchu były instruowane, aby postępować tak samo, przesyłając pakiet dalej. W rezultacie osoba końcowa, jeśli łańcuch nie pękł po drodze, otrzymywała od osoby, z którą jest „na ty”, przesyłkę wraz z listą osób pośrednich. Wyniki jednego z serii doświadczeń Milgrama były następujące: na 296 przesyłek 232 nie doszły do celu, co jest dość typową przypadłością badań socjologicznych. Natomiast spośród 64, które doszły (22%), znamienny był jeden fakt. Średnia (dokładniej mediana) liczby przesyłań wyniosła zaledwie 5.5. Inne doświadczenia, wielokrotnie powtarzane także przez innych badaczy, potwierdziły ten fakt. Zatem, aby dotrzeć do innej osoby poprzez łańcuch znajomości, wystarczy średnio 5-6 połączeń, co nazwano również „sześcioma stopniami oddalenia” (ang. *six-degree separation*): średnio, do prawie każdej osoby na świecie (prezydent Barack Obama, Doda, Tomasz Adamek, kasjerka w Tesco) można dotrzeć za pomocą zaledwie ok. sześciu „znajomościowych” kroków!

W języku teorii grafów sformułowalibyśmy problem następująco.

Hipoteza 5.1 (sześć stopni oddalenia). *Utwórzmy graf, którego wierzchołkami są ludzie, a krawędź łączy dwa wierzchołki wtedy i tylko wtedy, gdy te osoby są „na ty”. Dla prawie wszystkich losowo wybranych par wierzchołków tego grafu istnieje łącząca je droga o średniej długości ok. 6.*

²⁷ W Stanach Zjednoczonych jest to pokaźny krąg znajomych, bo w większości środowisk wszyscy z wszystkimi prawie natychmiast przechodzą „na ty”!

Doświadczenia Milgrama były realizacją wcześniejszych pomysłów sięgających roku 1929, kiedy Frigyes Karinthy²⁸ opublikował opowiadanie „Łańcuchy”, zawierające omawiane koncepcje sześciu stopni oddalenia i kurczącego się świata. Obecnie, w dobie internetu i globalnej wioski, doświadczenia nad sieciami społecznymi są intensywnie kontynuowane i poszerzane. Nie trzeba już wydawać pieniędzy na znaczki! Rysunek 5.39 pokazuje wynik takiego eksperymentu [50], gdzie 24000 ochotników próbowało skontaktować się za pomocą poczty elektronicznej z 18 wybranymi osobami-cełami. Dla 384 osób, którym kontakt się udał, wystarczyło zaledwie kilka kroków, ze średnią długością 4.01. Wyniki zdają się więc potwierdzać wnioski Milgrama. Bolączką tego typu doświadczeń jest jednak bardzo wysoki wskaźnik przerwania łańcucha, np. ktoś myśli, że to spam, nie ma czasu, z zasady nie wypełnia ankiet itp. W doświadczeniu z rys. 5.39 prawdopodobieństwo przerwania w każdym kroku wynosi aż 60%. W związku z tym jest bardzo mała szansa, aby długie łańcuchy wiadomości nie zostały przerwane. W eksperymencie [50] zaledwie ok. 1.6% zainicjowanych łańcuchów wiadomości osiągnęło swój cel, podczas gdy w doświadczeniu Milgrama było to „aż” 22%. Ten efekt jest podstawą uzasadnionej krytyki hipotezy 5.1 w podanym sformułowaniu. Prawdą jest, że dla tych wiadomości, które *osiągnęły cel*, wystarczyło kilka kroków, natomiast ich liczba wydaje się istotnie zależeć od prawdopodobieństwa przerwania łańcucha. Dokładniejsza analiza hipotezy wymagałaby więc dużej próbki danych i większej sumienności uczestników.

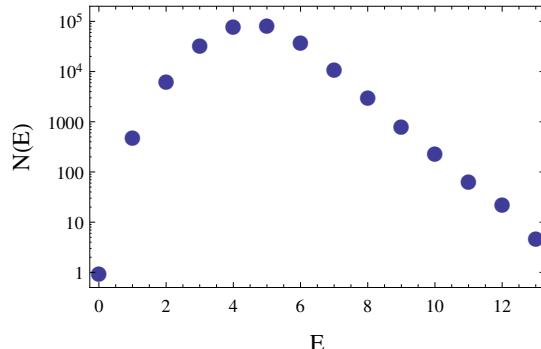
Bardzo ciekawym aspektem doświadczeń dotyczących „Małego świata” jest istota działania algorytmu znajdującego połączenie. Jest on oparty na „kolektywnej inteligencji” społeczeństwa. Ludzie, mający tylko lokalną wiedzę o swoich bezpośrednich znajomych, są w stanie przekazać w kilku krokach informację praktycznie do każdego mieszkańca Ziemi.

W społeczności matematyków odpowiednikiem sześciu stopni oddalenia jest tzw. *liczba Erdős'a*²⁹. Znany z ekscentrycznego zachowania Erdős współpracował z setkami matematyków, częstokroć w zamian za „wikt i opierunek”. W hołdzie koledzy zdefiniowali liczbę Erdős'a E w następujący sposób: sam Erdős ma $E = 0$, współautorzy jego publikacji $E = 1$, współautorzy współautorów niebędący współautorami Erdős'a $E = 2$ itd. Indukcyjnie, dany matematyk posiada $E = k$, jeśli ma wspólne publikacje z kimś o $E = k - 1$, a nie ma publikacji z nikim o $E \leq k - 2$.

Rysunek 5.40 pokazuje, ilu znamy matematyków o danej liczbie Erdős'a. Średnia tego rozkładu wynosi 4.65, a mediana 5, w całkowitej zgodności z hipotezą „sześciu stopni oddalenia”. Dane zostały skrupulatnie zebrane przy wykorzystaniu elektronicznych baz danych publikacji matematycznych, więc nie są obarczone problemem przerwania łańcucha komunikacji w oryginalnym eksperymencie Milgrama czy jego e-mailowej wersji. Według <http://www.oakland.edu/enp/>

²⁸ Frigyes Karinthy (1887-1938), węgierski pisarz i poeta.

²⁹ Paul Erdős (1913-1996), węgierski matematyk. Opublikował aż 1525 prac z kombinatoryki, teorii grafów, teorii mnogości, analizy matematycznej i rachunku prawdopodobieństwa.



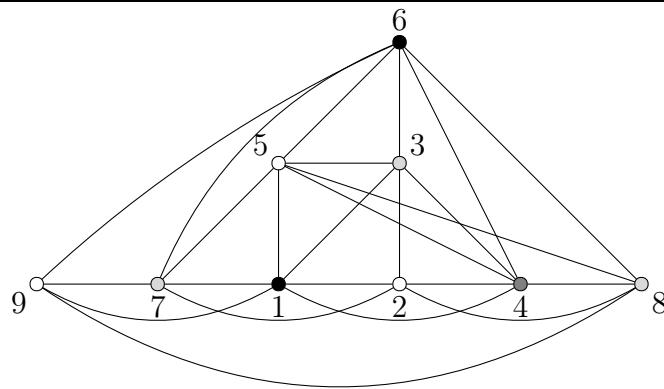
Rysunek 5.40: Rozkład liczby Erdősa $N(E)$ jest liczbą matematyków posiadających daną wartość E (dane z <http://www.oakland.edu/enp/>)

ok. 268000 matematyków ma liczbę Erdősa, 50000 opublikowało prace współautorskie, ale nie mają liczby Erdősa, ponadto 84000 opublikowało jedynie prace jednoautorskie.

5.12 Kolorowanie wierzchołkowe grafów

Wyobraźmy sobie następujący problem praktyczny: musimy ułożyć podział godzin dla studentów pierwszego roku informatyki na Uniwersytecie Jana Kochanowskiego w Kielcach. Studenci wybierają po kilka z oferowanych przedmiotów: 1) zagadnienia kolorowania grafów, 2) podstawy sztucznej inteligencji, 3) wprowadzenie do sieci społecznych, 4) ochrona drzew i lasów, 5) pracownia gier komputerowych, 6) historia buntu maszyn, 7) geografia Jawy, 8) lektorat *c++*, 9) aerobik w wodzie. Założymy, że zajęcia odbywają się w przedziałach czasowych 8:00-9:00, 9:00-10:00 itd. Pytanie jest następujące: *Jaka jest najmniejsza liczba przedziałów czasowych, aby móc ułożyć podział godzin w taki sposób, by wszyscy studenci mogli uczestniczyć w zajęciach, na które się zapisali?*

Niech owe dziewięć oferowanych przedmiotów stanowi wierzchołki grafu G . Z definicji, dwa wierzchołki są połączone krawędzią (są sąsiednie), jeśli istnieje student, który wybrał obydwa odpowiadające im przedmioty. W tej sytuacji zajęcia z tych przedmiotów nie mogą odbywać się jednocześnie, zatem krawędź oznacza relację „niemożliwej jednoczesności”. Przeglądamy zgłoszenia studentów i sporządzamy odpowiedni graf. Powiedzmy, że wygląda on w następujący sposób:



Powstały graf jest skomplikowany. Widzimy dużo połączeń, są jednak pary wierzchołków niesąsiednie, tj. niepołączone krawędziami, np. pary (1, 6) itd. Nikt nie zapisał się jednocześnie na „zagadnienia kolorowania grafów” i „historię buntu maszyn”. Następnie każdemu przedziałowi czasowemu przyporządkujemy umownie pewien kolor (na powyższym rysunku odcień szarości), np.:

- 8:00-9:00 – biały,
- 9:00-10:00 – czarny,
- 10:00-11:00 – ciemnoszary,
- 11:00-12:00 – jasnoszary itd.

Pokolorowanie wierzchołków grafu jest równoważne przyporządkowaniu danemu przedmiotowi przedziału czasowego. Jeśli pokolorujemy wierzchołki w taki sposób, że żadna sąsiednia para nie jest pokolorowana tym samym kolorem, to u żadnego studenta nie będzie konfliktu: będzie mógł uczęszczać na wszystkie zajęcia, na które się zapisał. Oczywiście 9 wierzchołków możemy zawsze pokolorować 9 kolorami, wtedy zajęcia będą trwać od 8:00 do 17:00, ale okazuje się, że wystarczy znacznie mniej kolorów. W rozważanym przez nas przykładzie wystarcza ich zaledwie cztery, jak zaznaczono na rysunku. Zajęcia można „upchać” między 8:00 a 12:00! A zatem problem bezkonfliktowego ułożenia podziału godzin rozwiązujemy, korzystając z teorii kolorowania grafów.

Są dwa podstawowe problemy związane z kolorowaniem grafów: ile kolorów potrzeba, aby pokolorować dany graf zgodnie z zadanimi regułami, oraz na ile sposobów można to zrobić. Najczęstszym sposobem kolorowania jest wykorzystane w powyższym przykładzie kolorowanie wierzchołkowe, określane w tym podrozdziale jako „kolorowanie” grafu.

Def. 5.35. *Graf jest k -barwny, jeśli za pomocą k kolorów można pomalować wierzchołki w taki sposób, że w wyniku sąsiednie wierzchołki nie mają tego samego koloru. Jeśli graf jest k -barwny, a nie jest $(k - 1)$ -barwny, to k nazywamy liczbą chromatyczną grafu G , oznaczaną jako $\chi(G)$, a graf nazywamy k -chromatycznym.*

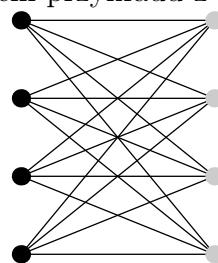
Zauważmy, że definicja „ k -barwności” określa, że k -kolorów wystarcza, ale nie mówi, że jest to najmniejsza potrzebna do pokolorowania liczba kolorów. Zatem graf czterobarwny jest również pięciobarwny, sześciobarwny itd. Natomiast k -chromatyczność jest właśnie najmniejszą potrzebną do pokolorowania liczbą kolorów. Pewne stwierdzenia dotyczące chromatyczności nieskomplikowanych grafów są natychmiastowe:

Tw. 5.15. Dla grafu pustego $\chi(N_n) = 1$, a dla grafu pełnego $\chi(K_n) = n$.

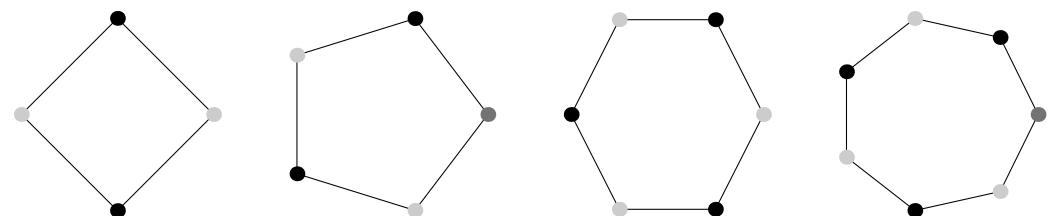
W oczywisty sposób dla grafu pełnego potrzebujemy tyle kolorów, ile wierzchołków. Gdyby kolorów było mniej, wtedy pewne dwa wierzchołki musiałyby być pomalowane tym samym kolorem, a ponieważ wszystkie pary wierzchołków są połączone, jest to niemożliwe.

Tw. 5.16. Niepusty graf G jest dwudzielny wtedy i tylko wtedy, gdy $\chi(G) = 2$.

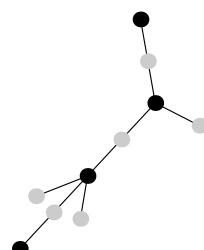
Dowód jest prostym uogólnieniem przykładu z poniższego rysunku.



Tw. 5.17. Dla grafu cyklicznego o parzystej liczbie wierzchołków mamy $\chi = 2$, a dla grafu cyklicznego o nieparzystej liczbie wierzchołków $\chi = 3$.



Tw. 5.18. Każde drzewo D jest 2-chromatyczne, $\chi(D) = 2$.



Tw. 5.19. *Jeśli s jest maksymalnym stopniem wierzchołka grafu G , to G jest $(s+1)$ -barwny.*

Dowód: (indukcja względem liczby wierzchołków) Założmy, że G ma n wierzchołków. Usuńmy na chwilę jeden z wierzchołków oznaczony jako w oraz przyległe doń krawędzie. Pozostały graf G' ma $n-1$ wierzchołków, a najwyższy stopień wynosi co najwyżej s – procedura usuwania mogła jedynie obniżyć stopień. Na mocy założenia indukcyjnego G' jest więc $(s+1)$ -barwny. Teraz dodajemy z powrotem, razem z usuniętymi krawędziami, wierzchołek w . Sąsiaduje on co najwyżej z s wierzchołkami, zatem kolorujemy go przy użyciu tego koloru, którego nie ma wśród sąsiadujących z nim s wierzchołków. Tak więc $s+1$ kolorów wystarcza na pokolorowanie grafu G . \square

Mocniejsze twierdzenie pochodzi od Brooksa³⁰:

Tw. 5.20 (Brooks (1941)). *Jeśli s jest maksymalnym stopniem wierzchołka spójnego prostego grafu G , to G jest s -barwny, z wyjątkiem grafu pełnego i grafu cyklicznego o nieparzystej liczbie wierzchołków, które są $s+1$ -barwne.*

Dowód pomijamy.

W zastosowaniu do naszego „grafu podziału godzin”, gdzie maksymalny stopień wierzchołka wynosi $s = 6$ (np. wierzchołek 6), powyższe twierdzenia pozwalają na ograniczenie liczby kolorów do siedmiu (Tw. 5.19) i do sześciu (Tw. 5.20), podczas gdy w rzeczywistości graf jest czterobarwny. Pokazuje to trudności, jakie napotykamy w zagadnieniach kolorowania grafów.

Nie jest znany szybki (o złożoności wielomianowej) algorytm kolorowania grafów. Poniżej przedstawiamy tzw. *algorytm z powracaniem*. Algorytmy tego typu są lepsze niż sprawdzanie wszystkich możliwości, ale ich czas wykonywania i tak jest długi, typowo rosnący wykładniczo z liczbą wierzchołków.

Alg. 5.11 (kolorowanie wierzchołkowe grafu z powracaniem). *Rozważ graf G o n wierzchołkach w_1, \dots, w_n oraz k kolorów.*

1. *Ponumeruj kolory od 1 do k .*
2. *Przydziel każdemu wierzchołkowi kolor 0 (brak koloru).*
3. *Podstaw $i = 1$, położ wierzchołek w_1 na stos roboczy.*
4. *Jeśli jest to możliwe, przypisz wierzchołkowi na wierzchu stosu najniższy kolor większy od jego bieżącego koloru i idź do kroku 6. W przeciwnym razie zdejmij wierzchołek z wierzchu stosu, nadaj mu kolor 0 i podstaw $i = i - 1$.*
5. *Jeśli stos jest pusty STOP – nie da się pokolorować grafu G przy użyciu k kolorów.*
6. *Jeśli pokolorowano już wszystkie wierzchołki STOP – mamy rozwiązanie.*

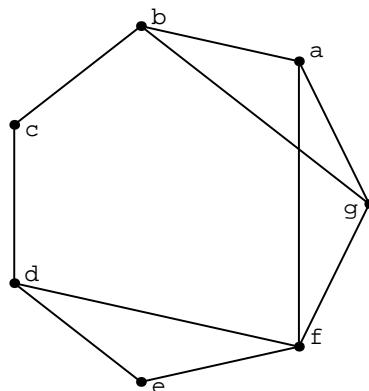
³⁰ Rowland Leonard Brooks (1916-1994), angielski matematyk.

Tabela 5.5: Działanie algorytmu 5.11 na grafie z rys. 5.41 dla $k = 3$ kolorów

Krok	a	b	c	d	e	f	g	Stos
1	1	0	0	0	0	0	0	a
2	1	2	0	0	0	0	0	ab
3	1	2	1	0	0	0	0	abc
4	1	2	1	2	0	0	0	abcd
5	1	2	1	2	1	0	0	abcde
6	1	2	1	2	1	3	0	abcdef
7	1	2	1	2	1	0	0	abcde
8	1	2	1	2	3	0	0	abcde
9	1	2	1	2	0	0	0	abcd
10	1	2	1	3	0	0	0	abcd
11	1	2	1	3	1	0	0	abcde
12	1	2	1	3	1	2	0	abcdef
13	1	2	1	3	1	2	3	abcdefg

7. Podstaw $i = i + 1$ i położ wierzchołek w_i na wierzch stosu. Wróć do kroku 4.

Przykład działania tego algorytmu zilustrowany jest na grafie z rys. 5.41, a poszczególne kroki ukazane są w tabeli 5.5. Zaczynamy od wierzchołka a , przyporządkowując mu kolor 1. Do kroku 6 algorytm „idzie do przodu”, jednak po tym kroku nie jesteśmy w stanie przyporządkować wierzchołkowi g żadnego z kolorów 1, 2, 3, ponieważ wierzchołek g jest sąsiadni do a , b i f . Wobec tego usuwamy f ze stosu (krok 7) oraz zwiększamy numer koloru wierzchołka e (krok 8). Kolor 2 nie jest możliwy, bo e sąsiaduje z d , więc przydzielimy e kolor 3. Próbujemy teraz ponownie iść do przodu, ale okazuje się, że nie możemy przydzielić f żadnego z trzech kolorów. Wobec tego wracamy do e . Ponieważ ma on już najwyższy kolor 3, usuwamy e ze stosu (krok 9). Zwiększamy teraz kolor wierzchołka d na 3 (krok



Rysunek 5.41: Przykładowy graf 3-chromatyczny, użyty do ilustracji algorytmu 5.11

10), po czym już bez cofania dochodzimy do końca algorytmu, znajdując rozwiązanie. Dla dwóch kolorów algorytm dochodzi do kroku 5 w tabeli 5.5, po czym musimy cofać się aż do usunięcia wszystkich wierzchołków ze stosu, co pokazuje, że dla $k = 2$ nie ma rozwiązania. Graf z rys. 5.41 jest więc 3-chromatyczny.

Najślynniejszym twierdzeniem dotyczącym kolorowania grafów jest niewątpliwie *twierdzenie o czterech kolorach*, mające bezpośredni związek z przedstawionym poniżej twierdzeniem o kolorowaniu map.

Tw. 5.21 (o czterech kolorach). *Każdy graf planarny jest czterobarwny.*

Twierdzenie to, chociaż wzmiankowane od połowy XIX w., zostało udowodnione przez Appela³¹ i Hakena³² dopiero w 1976 r. Co więcej, dowód jest zupełnie innej natury niż inne dowody przeprowadzane na przestrzeni uprzedniej historii matematyki: otóż jest to dowód komputerowy! Nie chodzi tu bynajmniej o heurystyczne sprawdzanie możliwości i obycie się z problemem, do czego komputer często nam służy. Dowód Appela i Hakena jest ścisły, a komputer potrzebny jest jedynie po to, aby sprawdzić olbrzymią, choć skończoną, liczbę przypadków. Znamienną rzeczą jest to, że człowiek nie byłby w stanie powtórzyć tego dowodu „na piechotę” ze względu na bardzo długi, dłuższy od czasu życia matematyka, czas obliczeń. Zastosowana metodologia przełamała więc pewną barierę i rozszerzyła praktyczne rozumienie, czym jest dowód w matematyce. Jednak wielu matematyków się zżyma: *jakie piękno można dostrzec w dowodzie długości książki telefonicznej? Czego można się nauczyć sprawdzając dowód poprzez sprawdzanie poprawności programu komputerowego?*

Problem kolorowania map jest następujący: *Ile najmniej kolorów potrzeba do pokolorowania mapy politycznej w taki sposób, aby dwa obszary posiadające granicę były pokolorowane różnymi kolorami?* (kraje stykające się wierzchołkiem mogą mieć ten sam kolor). Hipoteza czterech kolorów została postawiona przez Guthriego³³ i rozpowszechniona przez de Morgana³⁴. Zanim przedyskutujemy problem, zdefiniujmy pojęcie grafu dualnego.

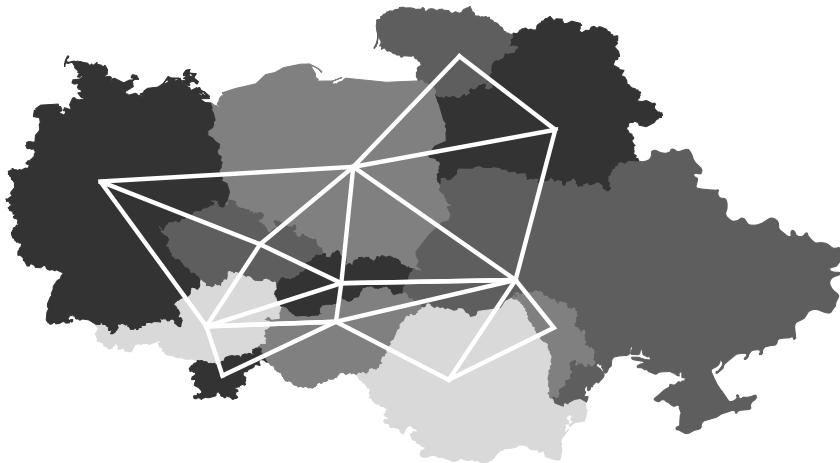
Def. 5.36. *Grafem dualnym do planarnego grafu G nazywamy graf G^* utworzony w następujący sposób: we wnętrzu każdej ściany grafu G wybieramy po jednym wierzchołku w_i^* . Zbiór $W^* = \{w_i^*\}$ jest zbiorem wierzchołków grafu G^* . Następnie dla każdej krawędzi e grafu G rysujemy linię przecinającą tę krawędź, która łączy dwa wierzchołki w_i^* i w_j^* leżące w ścianach po obu stronach krawędzi e . Tak skonstruowane linie są krawędziami grafu G^* .*

³¹ Kenneth Appel (1932–), amerykański matematyk.

³² Wolfgang Haken (1928–), amerykański matematyk.

³³ Francis Guthrie, południowoafrykański matematyk, student Augustusa de Morgana, zasłynął z wysuniętej w 1852 r. hipotezy o czterech kolorach, którą zauważył podczas kolorowania hrabstw na mapie Anglii.

³⁴ Augustus de Morgan (1806-1871), brytyjski matematyk i logik.



Rysunek 5.42: Mapa naszej części Europy i dorysowany odpowiadający jej graf dualny G^* . Graf wyjściowy G to graf, którego krawędzie są granicami państw, a wierzchołki punktami zbiegania się granic

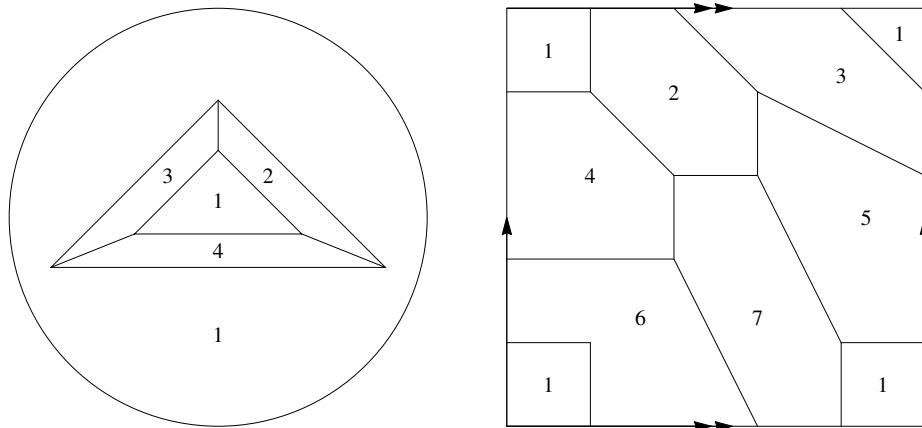
Przykład pokazano na rys. 5.42. Widzimy natychmiast, że w grafie dualnym są siedztwo wierzchołków oznacza graniczenie obszarów w oryginalnym grafie. Ponadto (dla grafów planarnych spójnych) graf dualny do dualnego jest izomorficzny do grafu wyjściowego, tzn. $(G^*)^* = G$. Możemy się o tym łatwo przekonać, rysując graf dualny do grafu G^* z rys. 5.42. Powstałe linie nowego można ułożyć wzdłuż granic państw, czyli krawędzi wyjściowego grafu G , otrzymując graf izomorficzny z G .

Ponieważ sąsiadowanie państw oznacza sąsiadowanie wierzchołków w grafie dualnym, problem kolorowania mapy jest równoważny dyskutowanemu wcześniej problemowi kolorowania wierzchołków grafu dualnego. Wobec tego Tw. 5.21 można sformułować w równoważny „kartograficzny” sposób:

Tw. 5.22 (o kolorowaniu map). *Każda mapę na płaszczyźnie (lub sferze) można pokolorować czterema kolorami w taki sposób, że graniczące państwa mają różne kolory (państwa stykające się wierzchołkowo mogą mieć ten sam kolor).*

Ponadto każde państwo musi być spójne, czyli wyłączamy sytuację, w której np. Rosję i obwód kaliningradzki musimy pokolorować tym samym kolorem. Dowód Tw. 5.22 wynika natychmiast z faktu, że graf dualny do każdej mapy jest planarny oraz z Tw. 5.21.

Ponoć kartografia miała niewielkie znaczenie dla rozwoju matematycznej teorii kolorowania map czy grafów. Większość map jest tradycyjnie kolorowana przy użyciu większej liczby kolorów niż cztery, a te, które używają czterech kolorów często mogą być pokolorowane jeszcze mniejszą ich liczbą. Przykład mapy, do pokolorowania której potrzebne są cztery kolory, pokazany jest w lewej części rys. 5.43.



Rysunek 5.43: Lewa strona: mapa na płaszczyźnie (lub sferze), do pokolorowania której potrzeba czterech kolorów. Prawa strona: mapa na torusie, do pokolorowania której potrzeba siedmiu kolorów. Aby powstał torus, należy skleić dolną i górną krawędź, tak aby pokryły się podwójne strzałki, a następnie zgiąć powstały „rulon” (w topologii zakładamy, że wszystko jest giętkie) w taki sposób, aby połączyć lewą i prawą krawędź mapy i pokryć pojedyncze strzałki. Wówczas cztery części „1” stają się jednym spójnym obszarem
(Parabola Online, http://www.parabola.unsw.edu.au/vol35_no2/node2.html)

Matematyczne uogólnienie problemu kolorowania to przypadek map narysowanych na powierzchniach o innych topologiach niż sfera, patrz rys. 5.24. Mapy rysowane na takich powierzchniach wymagają więcej kolorów niż na sferze. Na torusie potrzeba 7 kolorów, co ilustruje przykład z rys. 5.43. Większa liczba kolorów związana jest ze zwiększoną możliwością sąsiadowania dzięki topologicznym rączkom. Można pokazać, że w ogólności dla powierzchni zamkniętej liczba ta wynosi

$$p = \left\lceil \frac{7 + \sqrt{49 - 24\chi}}{2} \right\rceil, \quad (5.11)$$

gdzie $\chi = 2 - 2g$ jest charakterystyką Eulera powierzchni, a g jest liczbą „rączek” danej powierzchni topologicznej (por. rys. 5.24). W szczególności dla torusa $g = 1$, $\chi = 0$ oraz $p = 7$. Jedynym odstępstwem od wzoru (5.11) jest butelka Kleina³⁵, dla której $\chi = 0$, ale $p = 6$. Ze wzoru (5.11) wynika, że dla powierzchni o $g = 2$ mamy $p = 8$ itd.

Przykład kolorowania mapy na torusie przedstawiony jest w prawej części rys. 5.43. Widzimy, że dowolny z siedmiu obszarów styka się z każdym innym (pamiętamy o utożsamieniu lewego i prawego oraz górnego i dolnego boku kwadratu). Wobec tego graf dualny jest pełny i do jego pokolorowania potrzebujemy siedmiu kolorów (Tw. 5.15).

³⁵ Christian Felix Klein (1849-1925), niemiecki matematyk.

Do pokolorowania pewnych map kolorów może wystarczyć mniej niż liczba (5.11) wystarczająca do pokolorowania każdej mapy. Mamy w szczególności następujące twierdzenie o mapach dwubarwnych:

Tw. 5.23. *Mapa jest dwubarwna wtedy i tylko wtedy, gdy odpowiadający jej graf jest eulerowski.*

Twierdzenie to jest zilustrowane na rys. 5.44.

W ostatniej części tego podrozdziału przedyskutujemy zagadnienie kombinatoryczne związane z kolorowaniem wierzchołkowym grafów, mianowicie *na ile sposobów można pokolorować dany graf G przy użyciu k kolorów*. Liczbę tę oznaczamy jako $P_G(k)$. Okazuje się (patrz poniżej), że jest ona wielomianem zmiennej k .

Def. 5.37. *Wielomian chromatyczny grafu G , oznaczony jako $P_G(k)$, to liczba sposobów pokolorowania wierzchołkowego G za pomocą k kolorów.*

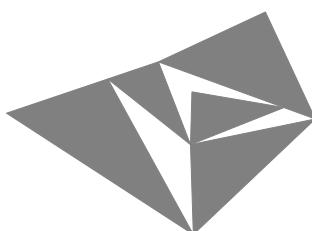
Kilka twierdzeń pozwala na skonstruowanie algorytmu znajdującego wielomian chromatyczny dla dowolnego grafu prostego.

Tw. 5.24. *Dla grafu pełnego K_n wielomian chromatyczny jest równy*

$$P_{K_n}(k) = k(k-1)\dots(k-n+1),$$

gdzie $k \geq n$.

Dowód: W grafie pełnym kolor każdego z n wierzchołków musi być różny. Pierwszy wierzchołek możemy pokolorować na k sposobów, drugi na $k-1$, trzeci na $k-2$, n -ty na $k-n+1$. Otrzymujemy więc liczbę n -elementowych wariacji bez powtórzeń k -elementowego zbioru kolorów, V_k^n . \square



Rysunek 5.44: Mapa dwubarwna odpowiadająca grafowi eulerowskiemu tworzonemu przez granice obszarów

Tw. 5.25 (fundamentalna redukcja). Niech v i w oznaczają niesąsiednie wierzchołki grafu prostego G . Utwórzmy graf G_1 poprzez dodanie krawędzi (v, w) , a także graf G_2 poprzez utożsamienie wierzchołków v i w oraz ewentualne utożsamienie krawędzi wielokrotnych, jeśli takie powstały. Wówczas zachodzi $P_G(k) = P_{G_1}(k) + P_{G_2}(k)$ [8, 51].

Dowód: Dowolne pokolorowanie grafu G nadaje wierzchołkom v i w albo ten sam kolor, albo różne kolory. Jeśli kolory są różne, to dodanie krawędzi (v, w) nie zmienia liczby pokolorowań, która wynosi $P_{G_1}(k)$. Jeśli kolor jest ten sam, to utożsamienie v z w nie zmienia liczby pokolorowań, wynoszącej $P_{G_2}(k)$. \square

Twierdzenie zilustrowane jest na rys. 5.45. Ponieważ grafy G_1 i G_2 są pełne, wobec tego znamy ich liczbę pokolorowań, patrz Tw. 5.24. Tak więc

$$P_G(k) = P_{G_1}(k) + P_{G_2}(k) = k(k-1)(k-2)(k-3) + k(k-1)(k-2) = k(k-1)(k-2)^2.$$

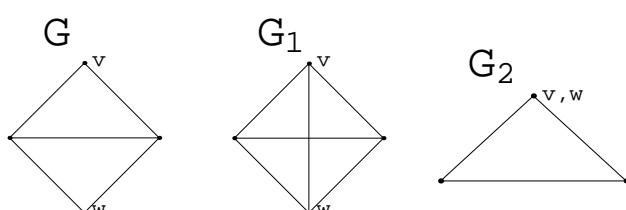
Wniosek 5.3. Zauważmy, że dopiero teraz mamy dowód, że „wielomian” chromatyczny jest w istocie wielomianem. Konstrukcję dodawania krawędzi lub utożsamiania wierzchołków możemy bowiem prowadzić tak długo, aż otrzymamy grafy pełne. Wtedy $P(G)$ jest sumą wielomianów dla grafów pełnych z Tw. 5.24.

Wniosek 5.4. Z postaci $P_G(k)$ wnioskujemy o chromatyczności grafu, najmniejsza bowiem liczba kolorów niezbędna do pokolorowania grafu to najmniejsza liczba k_{\min} dla której $P_G(k_{\min}) > 0$.

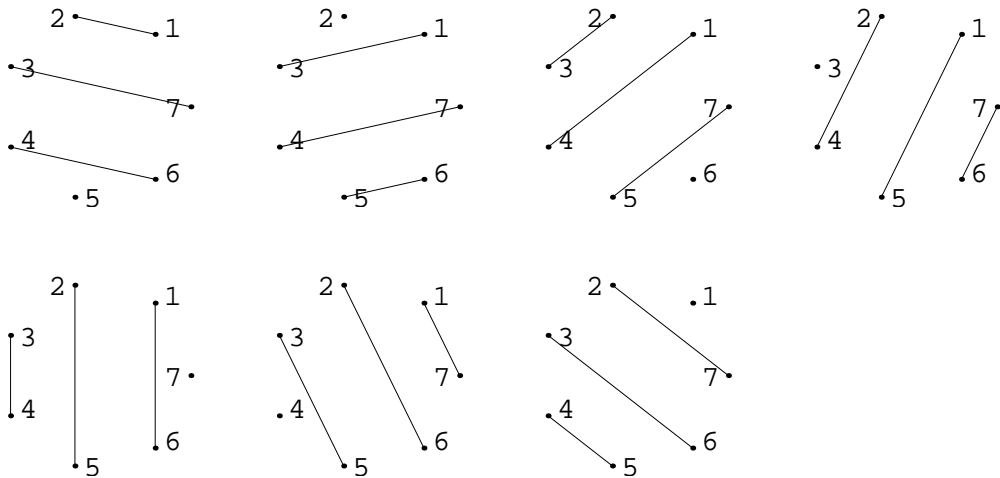
W szczególności graf G z rys. 5.45 jest 3-chromatyczny i $P_G(3) = 6$, tj. istnieje 6 sposobów pokolorowania tego grafu trzema kolorami.

5.13 Kolorowanie krawędziowe grafów

Zajmiemy się teraz innym sposobem kolorowania grafów, mianowicie *kolorowaniem krawędziowym*. Kolorowanie to polega na malowaniu krawędzi grafu w taki



Rysunek 5.45: Ilustracja Tw. 5.25. Liczba pokolorowań grafu G jest sumą liczb pokolorowań grafów G_1 i G_2

Rysunek 5.46: Kolejne rundy dzielenia się opłatkiem przez $n = 7$ osób

sposób, aby sąsiednie krawędzie nie miały tego samego koloru. Jako ilustrację zagadnienia rozważmy „problem dzielenia się opłatkiem”.

Problem 5.1. Na wigilii jest n osób. Jak czas trwania ceremonii dzielenia się opłatkiem zależy od n ?

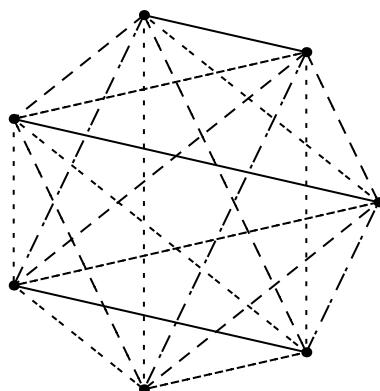
Niech składanie życzeń dla każdej pary wynosi t_o . Wszystkich życzeń jest oczywiście tyle, ile możliwych par, czyli $n(n - 1)/2$. Nie jest jednak prawdą, że cała ceremonia trwa $\frac{n(n-1)}{2}t_o$, ponieważ życzenia mogą być (i są) składane równocześnie. Zastanówmy się, używając teorii grafów, ile wynosi najkrótszy możliwy czas dzielenia się opłatkiem dla n osób, tzn. w maksymalny sposób wykorzystamy możliwość równoczesnego dzielenia się przez różne pary. Spójrzmy na rys. 5.46, ukazujący jedno z możliwych rozwiązań. Wierzchołki grafu przedstawiają osoby, a krawędzie składanie życzeń. W przykładzie osób jest 7. W pierwszej „rundzie” (lewy górny rysunek) mamy następujące pary składające sobie życzenia: 1-2, 3-7, 4-6. Osoba 7 nie ma pary i pauzuje. Kolejne rysunki przedstawiają następne rundy z innymi konfiguracjami par, w sumie rund jest 7. Widzimy, że w rezultacie każdy z każdym podzielił się opłatkiem, a cała „operacja” trwała $7t_o$.

Krawędziom w poszczególnych rundach możemy przydzielić różne kolory, w naszym przykładzie wyróżnione typem linii. Następnie nakładamy na siebie wszystkie 7 pokolorowanych rysunków 5.46, w wyniku czego dostajemy rys. 5.47. W wyniku konstrukcji każdy wierzchołek połączony jest z wszystkimi innymi, ale wszystkie krawędzie wychodzące z danego wierzchołka są różnokolorowe. Oznacza to, że każda osoba złożyła życzenia wszystkim pozostałym, ale w danym momencie (konkretny kolor krawędzi/typ linii) składała życzenia tylko jednej osobie. Wszystkie reguły zostały więc zachowane. Widzimy więc, że nasz problem

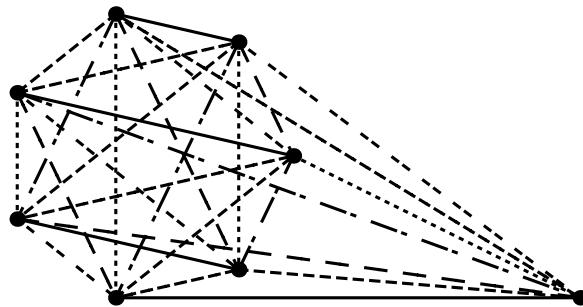
można w języku teorii grafów sformułować następująco: *Ile kolorów potrzeba do pokolorowania krawędziowego grafu pełnego K_n ?* Odpowiedź jest bardzo prosta.

Tw. 5.26. *Liczba kolorów potrzebnych do pokolorowania krawędziowego grafu pełnego o $n > 1$ wierzchołkach, K_n , wynosi n dla n nieparzystego i $n - 1$ dla n parzystego.*

Dowód: Dowód dla n nieparzystego już niemal zrobiliśmy. Wystarczy sobie uzmysłowić, że konstrukcja z rys. 5.47 jest ogólna, tj. można ją wykonać dla dowolnego n . Ustawiamy wierzchołki w wielokąt i kreślimy łączące je linie równoległe tak, jak na rys. 5.46. Tak więc ze wskazanej konstrukcji widać, że n kolorów wystarcza. Dla n parzystego wystarcza $n - 1$ kolorów. Dowód zilustrowany jest na rys. 5.48. Oddzielamy jeden wierzchołek i dla pozostałej nieparzystej liczby $n - 1$ wierzchołków wykonujemy poprzednią konstrukcję przy użyciu $n - 1$ kolorów. Następnie każdy z tych wierzchołków łączymy z oddzielonym wierzchołkiem takim kolorem, który nie został użyty do pokolorowania krawędzi wchodzących do tego wierzchołka. Zatem $n - 1$ kolorów wystarcza do pokolorowania krawędziowego K_n dla n parzystego. Oczywiście kolorów tych nie może być mniej, bo wówczas by ich zabrakło do pokolorowania $n - 1$ krawędzi wchodzących z każdego wierzchołka. Trzeba jeszcze pokazać, że dla n nieparzystego kolorów nie może być mniej niż n . Oddzielmy jeden wierzchołek v . Pozostały graf P o parzystej liczbie $n - 1$ wierzchołków wymaga, jak pokazaliśmy powyżej, co najmniej $n - 2$ kolorów do pokolorowania. Z każdego wierzchołka grafu P wychodzi $n - 2$ różnokolorowych krawędzi. Połączmy teraz oddzielony wierzchołek v z grafem P . Łącząc v z dowolnym wierzchołkiem P , potrzebujemy kolejnego koloru, o numerze $n - 1$, co kończy dowód. \square



Rysunek 5.47: Wynik nałożenia pokolorowanych grafów z rys. 5.46. Kolory, odpowiadające kolejnym rundom składania życzeń, zastąpione są tutaj typem linii. Każdy wierzchołek połączony jest z wszystkimi innymi oraz wszystkie krawędzie wychodzące z danego wierzchołka są różnokolorowe (mają inny typ linii)



Rysunek 5.48: Ilustracja dowodu, że do pokolorowania grafu pełnego o parzystej liczbie wierzchołków n wystarcza $n - 1$ kolorów

Wracając jeszcze na chwilę do problemu opłatkowego, zauważmy, że pewien czas pomiędzy kolejnymi życzeniami tracony jest na poszukiwanie osób, z którymi jeszcze się nie łamaliśmy. Czas ten jest z grubsza proporcjonalny do n , co zapiszemy jako nt_s . Koniec końców całkowity czas ceremonii wynosi $t = nt_s + nt_o$ dla n nieparzystego oraz $t = nt_s + (n - 1)t_o$ dla n parzystego, a dla dużych n możemy po prostu napisać $t \sim n$. Doświadczenie potwierdza teorię. Życzenia wigilijne w domu zazwyczaj zajmują parę minut, podczas gdy uroczysty opłatek u JM Rektora (gdzie gości jest rzędu 20 razy więcej) trwa ok. dwie godziny...

5.14 Liczby Ramseya

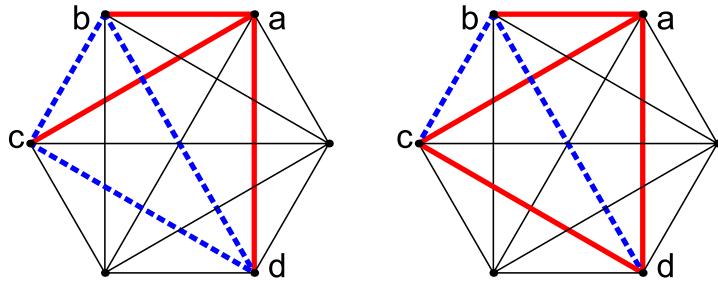
Z kolorowaniem krawędzi łączy się następujące znane twierdzenie:

Tw. 5.27 (o obcych i przyjaciołach). *Na przyjęciu jest grupa n osób. Jeśli dwie osoby spotkały się wcześniej, określamy ich jako znajomych, w przeciwnym razie jako nieznajomych. Jeśli $n \geq 6$, to wśród gości mamy zawsze przynajmniej 3 osoby, które się parami znają (tj. każdy z każdym), lub 3 osoby, które się parami nie znają.*

Spróbujmy przeformułować twierdzenie w języku grafów. Jeśli osoby (wierzchołki) się znają, to łączymy je krawędzią czerwoną (linia ciągła), w przeciwnym razie krawędzią niebieską (linia przerywana). Ponieważ każde dwie osoby albo się znają, albo nie, powstaje w ten sposób graf pełny z krawędziami pokolorowanymi dwoma kolorami. Możemy więc zapisać alternatywnie:

Tw. 5.28. *Każdy graf pełny o $n \geq 6$ wierzchołkach pokolorowany dwoma kolorami zawiera podgraf pełny o trzech wierzchołkach, mający krawędzie tego samego koloru.*

Dowód: Z wierzchołka a wychodzi 5 krawędzi. Na mocy zasady szufladkowej, co najmniej 3 są tego samego koloru (przyjmijmy, że czerwonego – linie ciągłe).



Rysunek 5.49: Ilustracja dwóch sytuacji z dowodu Tw. 5.28. W obydwu powstaje trójkąt o bokach tego samego koloru (oznaczonego typem linii). Cienkie linie odpowiadają krawędziom o dowolnym kolorze (jednym z dwóch)

Niech prowadzą one do wierzchołków b , c i d . Jeśli krawędzie (b, c) , (c, d) i (d, b) są niebieskie (linie przerwywane), to tworzą niebieski trójkąt i mamy tezę. Jeśli nie, to któraś z nich jest czerwona. Ponieważ jej końce są połączone z wierzchołkiem a czerwonymi krawędziami, tworzą czerwony trójkąt i ponownie odnajdujemy tezę. \square

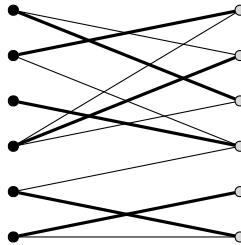
Dowód zilustrowany jest na rys. 5.49.

Z powyższym zagadnieniem związane są *liczby Ramseya*³⁶, zdefiniowane następująco:

Def. 5.38. *Liczba Ramseya $R(k, l)$ oznaczamy takie najmniejsze n , dla którego graf K_n pokolorowany krawędziowo dwoma kolorami zawiera graf K_k o krawędziach jednego koloru lub graf K_l o krawędziach drugiego koloru. Z definicji, $R(k, l) = R(l, k)$.*

Twierdzenie 5.28 razem z wynikiem ćw. 22 oznacza, że $R(3, 3) = 6$. W ogólności okazuje się, że znajdowanie liczby Ramseya jest zadaniem niezwykle złożonym obliczeniowo. W szczególności w algorytmie brutalnej siły mamy $2^{n(n-1)/2}$ możliwości pokolorowania krawędzi grafu pełnego K_n dwoma kolorami i ich przeglądnięcie nawet dla niewielkich wartości n jest niemożliwe. Tabelę znanych liczb Ramseya można znaleźć w [52]. W szczególności liczba $R(5, 5)$ nie jest znana dokładnie, mamy tylko ograniczenie $43 \leq R(5, 5) \leq 49$.

³⁶ Frank Plumpton Ramsey (1903-1930), matematyk, brytyjski filozof i ekonomista brytyjski.



Rysunek 5.50: Przykład maksymalnego skojarzenia (oznaczonego wytłuszczeniem) w grafie dwudzielnym

5.15 Kojarzenie małżeństw

Kolejnym bardzo znanym problemem teorii grafów, którego znaczenie wykraca poza sprawy matrymonialne sugerowane w nazwie, jest tzw. *kojarzenie małżeństw* [8].

Problem 5.2 (problem kojarzenia małżeństw). *Rozważmy zbiór n kobiet i n mężczyzn, w którym mamy pewne znajomości. W jakich warunkach każdej kobiecie można zaproponować kandydata na męża (tj. mężczyznę, którego zna)? Kandydaci nie mogą się powtarzać, tzn. dany mężczyzna może być zaproponowany tylko jednej kobiecie.*

Aby sformułować problem w języku grafów, potrzebna jest definicja maksymalnego skojarzenia.

Def. 5.39. *Maksymalnym skojarzeniem w grafie dwudzielnym o rozłącznych zbiorach wierzchołków V_1 i V_2 nazywamy taki podzbiór krawędzi, dla którego każdy wierzchołek V_1 oraz każdy wierzchołek V_2 jest końcem dokładnie jednej krawędzi.*

Przykład pokazany jest na rys. 5.50. A oto problem kojarzenia małżeństw dla grafów:

Problem 5.3. *W jakich warunkach istnieje maksymalne skojarzenie w grafie dwudzielnym?*

Odpowiedź, jak się okazuje niezwykle prosta choć nieoczywista, zawarta jest w Tw. Halla³⁷:

Tw. 5.29 (Halla o małżeństwach). *Problem kojarzenia małżeństw ma rozwiązanie wtedy i tylko wtedy, gdy każdy podzbiór $k \leq n$ kobiet zna przynajmniej k mężczyzn.*

³⁷ Philip Hall (1904-1982), brytyjski matematyk znany z prac nad teorią grup.

Dowód: Jeśli istnieje maksymalne skojarzenie, to oczywiste jest, że każdy podzbiór k kobiet zna k mężczyzn. Każda kobieta zna bowiem mężczyznę, z którym jest swatana. Dowód twierdzenia w drugą stronę przebiega indukcyjnie. Twierdzenie jest trywialnie prawdziwe dla $n = 1$. Założymy, że jest prawdziwe dla wszystkich $m \leq n$ i rozważmy zbiór $n + 1$ kobiet i mężczyzn. Wyróżnimy dwie możliwe sytuacje: (a) Każdy podzbiór $k < n + 1$ kobiet zna przynajmniej $k + 1$ mężczyzn, tzn. jeden z nich pozostaje „zapasowym narzeczonym”. Wówczas swatamy dowolną parę i wyłączamy ją z dalszych rozważań. Pozostała grupa n kobiet spełnia założenie, że każdy jej k -elementowy podzbiór zna co najmniej k mężczyzn (wcześniej знаły przynajmniej $k + 1$ mężczyzn, ale jednego „utraciły” na rzecz wyswatanej właśnie koleżanki). Spełnione jest zatem założenie twierdzenia dla n i możemy pozostałe kobiety również wyswatać. (b) W drugiej sytuacji istnieje podzbiór $k < n + 1$ kobiet znających dokładnie k mężczyzn. Wówczas z założenia indukcyjnego możemy ten podzbiór wyswatać. Pozostałe $n+1-k \leq n$ kobiet również spełnia założenie indukcyjne. Gdyby tak nie było, pewien podzbiór l z nich znałby mniej niż l niewyswatanych mężczyzn. Dołączając do tego podzbioru zbiór już wyswatanych kobiet otrzymalibyśmy warunek, że $l+k$ kobiet znałoby mniej niż $l+k$ mężczyzn, co przeczy założeniu odnośnie do całego zbioru $n + 1$ kobiet. \square

Jako ilustrację części (b) dowodu możemy wykorzystać rys. 5.50, gdzie $n + 1 = 6$. Cztery kobiety (cztery górne wierzchołki po lewej stronie grafu) znają dokładnie czterech mężczyzn (górnne wierzchołki po prawej stronie). Pozostałe dwie kobiety muszą znać pozostałykh kawalerów. Gdyby знаły tylko jednego spośród nich, łącznie z wyswatana już grupą sześć kobiet znałoby tylko pięciu mężczyzn, co jest w sprzeczności z założeniem.

Następujący przykład z kartami przekona nas, że Tw. 5.29 nie jest trywialne. Weźmy talię 52 kart i potasowawszy rozłożmy ją na 13 kupek po 4 karty. Czy w każdym przypadku można wybrać z każdej kupki po jednej karcie w taki sposób, abyśmy mieli 13 kart, po jednej z każdego waloru, tj. as, król, dama, ..., trójka, dwójka (karty mogą być różnego koloru)? Odpowiedź nie jest na pierwszy rzut oka oczywista. Eleganckiej odpowiedzi dostarcza Tw. Halla. Niech 13 walorów będzie kobietami, a 13 kupek mężczyznami. Każda kobieta zna jednego mężczyznę, bo np. jakiś as znajduje się w którejś z kupek, jakiś król też itd. Rozważmy teraz dwa walory, np. as i król. Ponieważ kart tych jest 8, muszą być rozłożone w co najmniej dwóch kupkach. Tak więc każda „kobieta” zna przynajmniej dwóch „mężczyzn”. Jeśli weźmiemy k walorów, to odpowiednie $4k$ karty muszą być w przynajmniej k kupkach. Tak więc spełnione są założenia Tw. 5.29.

Przedstawiony dowód Tw. 5.29 nie jest konstrukcyjny, więc nie pokazuje, w jaki sposób znaleźć rozwiązanie. Zastosowanie znajduje tu np. algorytm z powracań analogiczny do algorytmu 5.11:

Alg. 5.12 (swatanie). *Mamy graf dwudzielny o n kobietach i n mężczyznach.*

1. Ponumeruj kobiety od 1 do n , zainicjalizuj $i = 1$.
2. Jeśli to możliwe
 - Przydziel kobiecie i jednego z jej znajomych, do tej pory z nią nieswanego. Wyjmij tego mężczyznę z grupy dostępnych kawalerów dla innych kobiet.
 - Dotychczasowego narzeczonego kobiety i przywrócić do puli dostępnych kawalerów dla innych kobiet.
 - Jeśli $i = n$ STOP – mamy maksymalne skojarzenie.
 - Podstaw $i = i + 1$.
3. W przeciwnym wypadku cofnij się, podstawiając $i = i - 1$. Jeśli $i = 1$ STOP – graf nie spełnia założeń Tw. Halla.
4. Idź do kroku 2.

Bardziej praktyczne zastosowanie Tw. Halla pojawia się przy ewaluacji wykonalności projektu: czy n „fachowców” jest w stanie wykonać n zadań w danym projekcie? Wyobraźmy sobie, że Instytut Fizyki potrzebuje wykładowców dla swoich kursowych przedmiotów (każdy profesor prowadzi w semestrze tylko jeden wykład, ale w ogólności zna się na $l \geq 1$ przedmiotach, które mógłby wykładać). Należy w tym celu sprawdzić, czy każda grupa k profesorów może prowadzić k przedmiotów.

Zauważmy jednak, że sprawdzanie „na piechotę” warunków Tw. Halla jest czasochłonne, ponieważ mamy aż $2^n - 1$ niepustych podzbiorów zbioru kobiet. W podrozdziale 5.19 poznamy metodę znajdującą maksymalne skojarzenie w czasie $\mathcal{O}(n^3)$.

5.16 Drzewa etykietowane

W tym podrozdziale omówimy bardzo ważne zastosowanie drzew, służące do przechowywania i szybkiego pozyskiwania danych.

Def. 5.40. Drzewem z wyróżnionym korzeniem nazywamy drzewo, w którym jeden wierzchołek, nazywany korzeniem, jest wyróżniony.

Drzewa takie piszemy zazwyczaj z korzeniem na górze, a na kolejnych poziomach poniżej wypisujemy kolejne pokolenia. Dwa wierzchołki na sąsiednich poziomach połączone krawędzią nazywamy *rodzicem* (wierzchołek powyżej) i *dzieckiem* (wierzchołek poniżej). Każdy wierzchołek oprócz korzenia ma dokładnie jednego rodzica, natomiast jeden rodzic może mieć kilkoro dzieci. Wierzchołki, które nie mają dzieci, nazywamy *liśćmi* (patrz rys. 5.51). Wierzchołki niebędące liśćmi nazywamy *węzłami gałęzi*. Ponadto wierzchołek p jest potomkiem wierzchołka v , jeżeli $v \neq p$ oraz v należy do drogi prostej od r do p . Wysokością h drzewa nazywamy numer najniższego poziomu.

Def. 5.41. Drzewem o $n \geq 2$ rozgałęzieniach nazywamy drzewo, w którym każdy rodzic ma co najwyżej n dzieci. Dzieci te oznaczamy liczbami ze zbioru $\{1, 2, \dots, n\}$. Jeśli dla danego rodzica numer i nie jest użyty, to mówimy, że dziecko i jest nieobecne.

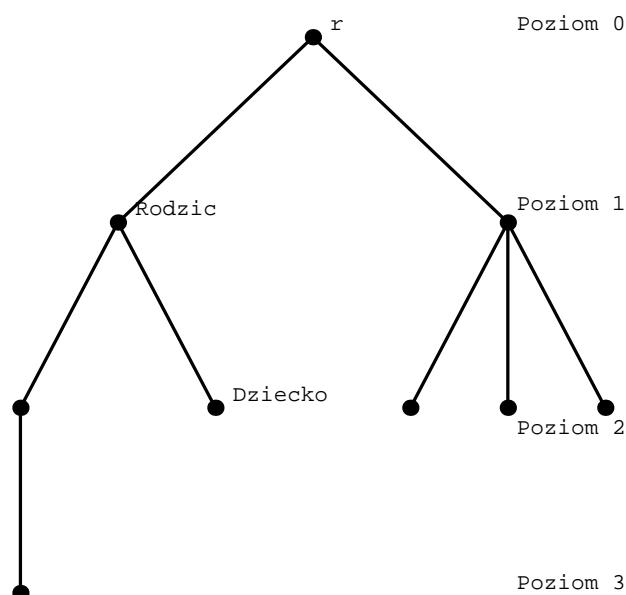
Def. 5.42. Drzewem binarnym nazywamy drzewo o dwóch rozgałęzieniach. Każdy rodzic może mieć co najwyżej dwoje dzieci: dziecko lewe, dziecko prawe, oboje dzieci, lub żadnego.

Def. 5.43. Drzewo o n rozgałęzieniach nazywamy regularnym, jeśli każdy rodzic ma n lub zero dzieci.

Inaczej, każdy wierzchołek ma albo największą możliwą liczbę dzieci, albo jest liściem.

Def. 5.44. Drzewo regularne o n rozgałęzieniach jest pełne, jeżeli wszystkie liście znajdują się na tym samym poziomie.

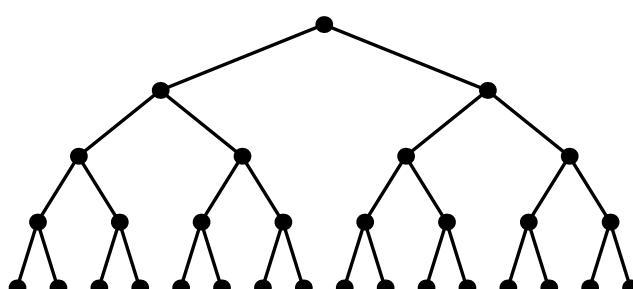
Oczywiście poziom ten jest zarazem wysokością drzewa. Przykład drzewa pełnego przedstawiony jest na rys. 5.52. Łatwo policzyć, że dla drzewa pełnego o n rozgałęzieniach i wysokości h mamy $L = n^h$ liści oraz $R = (n^h - 1)/(n - 1)$ rodziców, co łącznie daje $N = (n^{h+1} - 1)/(n - 1)$ wierzchołków. Dla drzewa binarnego $n = 2$, więc $L = 2^h$, $R = 2^h - 1$ i $N = L + R = 2^{h+1} - 1$. W szczególności drzewo z rys. 5.52 o $h = 4$ ma $L = 2^4 = 16$ liści i $R = 15$ rodziców, łącznie $N = 31$ wierzchołków.



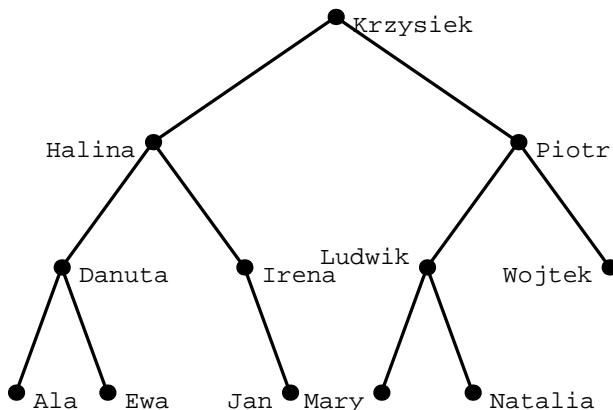
Rysunek 5.51: Drzewo z wyróżnionym korzeniem r

Powyżej zdefiniowane drzewa używane są do przechowywania danych. W tym celu każdemu wierzchołkowi przyporządkowujemy *etykietę*, która de facto niesie jakąś informację, którą pragniemy zapisać. Przykład takiego *drzewa etykietowanego* pokazany jest na rys. 5.53. Drzewo takie nazywane jest *drzewem poszukiwań binarnych*. W tym przypadku drzewo zawiera strukturę danych (imiona) uporządkowanych w sposób alfabetyczny. Aby zrozumieć celowość takiej organizacji danych, wyobraźmy sobie, że szczęśliwie skończyliśmy już studia i prowadzimy firmę. Naszym pierwszym klientem jest Krzysiek, który staje się korzeniem drzewa. Oczywiście, odpowiedni rekord zawiera niezbędne informacje o Krzyśku jako kliencie: adres, telefon itd. Następną klientką, która pojawiła się w firmie, jest Halina. Ponieważ alfabetycznie H poprzedza K, jej rekord umieszczamy na lewo od Krzyśka, oczywiście na poziomie 1. Halina staje się więc *lewym dzieckiem* Krzyśka. Następnie pojawia się Irena, która alfabetycznie poprzedza Krzyśka, ale następuje po Halinie. Jest więc na lewo od Krzyśka, idziemy więc po gałęzi do Haliny, natomiast musi być od niej na prawo, schodzimy więc na poziom 2 i zapisujemy Irenę jako prawe dziecko Haliny. Konstrukcja struktury jest zatem następująca: przy pojawianiu się kolejnego klienta X porównujemy jego imię z imieniem korzenia. Jeśli alfabetycznie poprzedza imię korzenia, to idziemy w dół w lewo, jeśli alfabetycznie następuje po nim, to idziemy w dół w prawo. Jeśli miejsce jest wolne, tworzymy nowy wierzchołek i zapisujemy w nim rekord klienta X . Jeśli miejsce jest zajęte przez już istniejący wierzchołek v , to dokonujemy kolejnego porównania kolejności alfabetycznej i idziemy w dół albo w prawo, albo w lewo, w zależności czy imię X poprzedza, czy następuje po imieniu v . Procedurę powtarzamy iteracyjnie aż do znalezienia wolnego miejsca. Założymy, że mamy już drzewo jak na rys. 5.53 i pojawił się klient o imieniu Mietek. Jest on więc na prawo od Krzyśka, na lewo od Piotra, na prawo od Ludwika i na lewo od Natalii. Odpowiadający Mietkowi wierzchołek powstanie więc na poziomie 4 jako lewe dziecko Natalii.

Przy odczytywaniu rekordów postępujemy analogicznie. Na przykład, aby dostać się do W , potrzebujemy dwóch porównań: z Krzyśkiem i Piotrem.



Rysunek 5.52: Drzewo binarne pełne o wysokości 4



Rysunek 5.53: Drzewo poszukiwań binarnych

Podstawową zaletą organizacji danych w postaci drzewa poszukiwań binarnych jest szybkość dotarcia do danego rekordu. Gdyby drzewo było drzewem pełnym (rys. 5.52), wówczas potrzebowalibyśmy co najwyżej h porównań, aby dotrzeć do danego wierzchołka, przy czym w tym przypadku $h = \log_2 \frac{N+1}{2}$ (dla drzewa z rys. 5.52 $h = 4$ i $N = 31$). W typowym przypadku, gdy drzewo poszukiwań binarnych nie jest pełne, również potrzebujemy liczby porównań rzędu $\log_2 N$, gdzie N jest liczbą wierzchołków. Jest to bardzo wolno rosnąca z N liczba, zatem algorytm pozwala na bardzo szybkie dotarcie do interesujących nas danych nawet w przypadku dużych N .

Omówimy teraz zagadnienie przeszukiwania drzew z wyróżnionym korzeniem. Trzy przedstawione poniżej algorytmy są rekurencyjnymi algorytmami przeszukiwania w głąb (patrz podrozdział 5.7) i jako wynik tworzą listę wszystkich wierzchołków w odpowiedniej (zależnej od algorytmu) kolejności. Pierwszym z nich jest algorytm tworzenia *prefiksowej* listy wierzchołków, tzn. takiej listy, w której rodzice poprzedzają swoje dzieci.

Alg. 5.13 (tworzenie listy prefiksowej).

A. Zdefiniuj procedurę $\text{Pre}(x) = \{1. \text{ Dopełnij } x \text{ do końca listy } L. 2. \text{ Jeśli } x \text{ jest liściem, wyjdź. 3. Dla każdego dziecka } v \text{ wierzchołka } x, \text{ od lewej do prawej strony, wykonaj rekurencyjnie } \text{Pre}(v).\}$

B. Wykonaj $\text{Pre}(r)$, gdzie r jest korzeniem drzewa.

Prześledźmy działanie algorytmu 5.13 na przykładzie drzewa z rys. 5.53, pisząc w nawiasie pierwsze litery etykiet wierzchołków umieszczonych na liście, czyli Krzysiek=K itd. Startujemy z korzenia ($L = K$), idziemy w lewo w dół ($L = KH$), znowu w lewo w dół ($L = KHD$) i jeszcze raz w lewo w dół ($L = KHDA$). A nie ma dzieci, więc idziemy do następnego dziecka wierzchołka D , czyli E ($L = KHDAE$). Ponieważ D nie ma więcej dzieci, idziemy do kolejnego dziecka H , czyli I ($L = KHDAEI$). Postępując do końca, otrzymujemy

ostateczną listę prefiksową grafu, $L = KHDAEIJPLMNW$. Opisana procedura przeszukiwania grafu zaznaczona jest na rys. 5.54. Po prostu obchodzimy graf na ok. wzdłuż zaznaczonej linii przerywanej, zaczynając z korzenia w kierunku odwrotnym do ruchu wskazówek zegara. Przy pierwszych odwiedzinach każdego wierzchołka wpisujemy go na listę. W ten sposób rodziniec wpisywany jest zawsze przed swoimi dziećmi.

Drugi algorytm przeszukiwania tworzy listę *postfiksową*, czyli listę, w której dzieci poprzedzają swoich rodziców.

Alg. 5.14 (tworzenie listy postfiksowej).

A. Zdefiniuj procedurę $\text{Post}(x) = \{1. \text{ Utwórz pustą listę } L(x). 2. \text{ Jeśli } x \text{ jest liściem, wyjdź. 3. Dla każdego dziecka } v \text{ wierzchołka } x, \text{ od lewej do prawej strony, wykonaj rekurencyjnie } \text{Post}(v) \text{ i dołącz otrzymaną listę } L(v) \text{ na końcu listy } L(x). 4. \text{ Dopusz wierzchołek } x \text{ na końcu listy } L(x).\}$

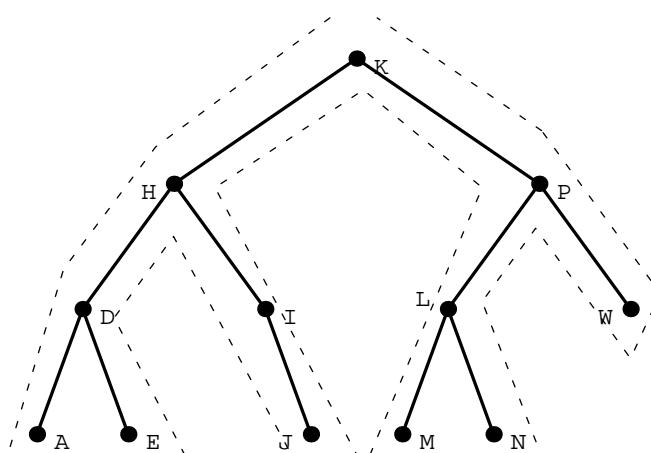
B. Wykonaj $\text{Post}(r)$, gdzie r jest korzeniem drzewa.

Do ilustracji działania algorytmu 5.14 posłużymy się ponownie rys. 5.54, znowu obchodzonym na ok. wzdłuż linii przerywanej. Tym razem jednak, w odróżnieniu od algorytmu 5.13, wpisujemy wierzchołek na koniec listy L podczas jego ostatnich odwiedzin, niejako „na pożegnanie”. W ten sposób powstaje lista postfiksowa wierzchołków $L = AEDJIHMNLPWK$.

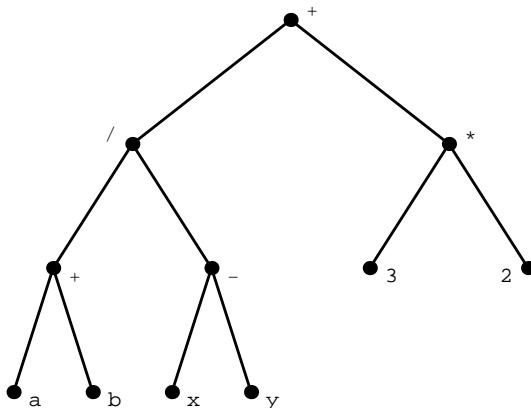
Ostatni algorytm dotyczy tylko drzew *binarnych*, dla których można utworzyć tzw. *infiksową* listę wierzchołków. W tej liście lewe dziecko poprzedza swojego rodzica, a dziecko prawe następuje po nim.

Alg. 5.15 (tworzenie listy infiksowej).

A. Zdefiniuj procedurę $\text{In}(x) = \{1. \text{ Utwórz pustą listę } L(x). 2. \text{ Jeśli } x \text{ jest liściem, wyjdź. 3. Jeśli wierzchołek } x \text{ ma lewe dziecko } l, \text{ wykonaj rekurencyjnie}\}$



Rysunek 5.54: Przeszukiwanie drzewa celem tworzenia listy prefiksowej lub postfiksowej



Rysunek 5.55: Drzewo przedstawiające wyrażenie algebraiczne $(a + b)/(x - y) + 3 \cdot 2$

In(l) iłącz otrzymaną listę $L(l)$ na początku listy $L(x)$. 4. Dołącz wierzchołek x do końca listy $L(x)$. 5. Jeśli wierzchołek x ma prawe dziecko p , wykonaj rekurencyjnie In(p) iłącz otrzymaną listę $L(p)$ na końcu listy $L(x)$.

B. Wykonaj In(r), gdzie r jest korzeniem drzewa.

Wykonanie tego algorytmu na drzewie z rys. 5.53 tworzy następującą infiksową listę wierzchołków: $L = ADEHIJKLMNOPW$. Graficznie można sobie wyobrazić tworzenie tej listy poprzez opuszczenie „pionowo w dół” na najniższy poziom rysunku wszystkich etykiet.

5.17 Notacja polska

Łukasiewicz³⁸ dokonał niezwykle ważnej obserwacji: *kazde wyrażenie algebraiczne zawierające działania jedno- i dwuargumentowe można zapisać bez użycia nawiasów!* W tym celu wykorzystuje się właśnie listy prefiksowe lub postfiksowe.

Rozważmy przykładowe wyrażenie $(a + b)/(x - y) + 3 \cdot 2$. Notacja ta jest notacją infiksową: operator działania napisany jest pomiędzy swoimi argumentami, np. $a + 2$, $x - y$ itd. Oczywiście potrzebne są tu nawiasy – inaczej, jak doskonale wiemy, wyrażenie nie byłoby jednoznacznie określone. Możemy nasze wyrażenie przedstawić w postaci drzewa z wyróżnionym korzeniem, tak jak na rys. 5.55. Liśćmi są tu liczby, a wierzchołkami działania dwuargumentowe. Obliczenie wyrażenia polega na wykonaniu operacji, począwszy od najniższego poziomu drzewa, a skończywszy na najwyższym. Dla przypadku $a = 1$, $b = 2$, $x = 4$, $y = 3$ procedurę obliczania przedstawiono na rys. 5.56.

Widzimy więc, że tak naprawdę wyrażeniu algebraicznemu odpowiada pewne drzewo binarne, w którym liśćmi są liczby, a węzły gałęziami operacjami algebraicznymi. W szkole poświecono bardzo wiele czasu, aby wpoić nam notację infiksową wyrażeń algebraicznych i większość z nas tylko w ten sposób może

³⁸ Jan Łukasiewicz (1878-1956), polski logik i filozof, współtwórca lwowsko-warszawskiej szkoły matematyki, jeden z najwybitniejszych logików XX w.

o nich „myśleć”. Notacja infiksowa jest jednak tylko jedną z trzech naturalnych możliwości, a konieczność używania nawiasów powoduje, że jest znacznie mniej wygodna do zastosowań informatycznych. Drzewo z rys. 5.55, a tym samym nasze wyrażenie algebraiczne, można przedstawić w postaci listy prefiksowej

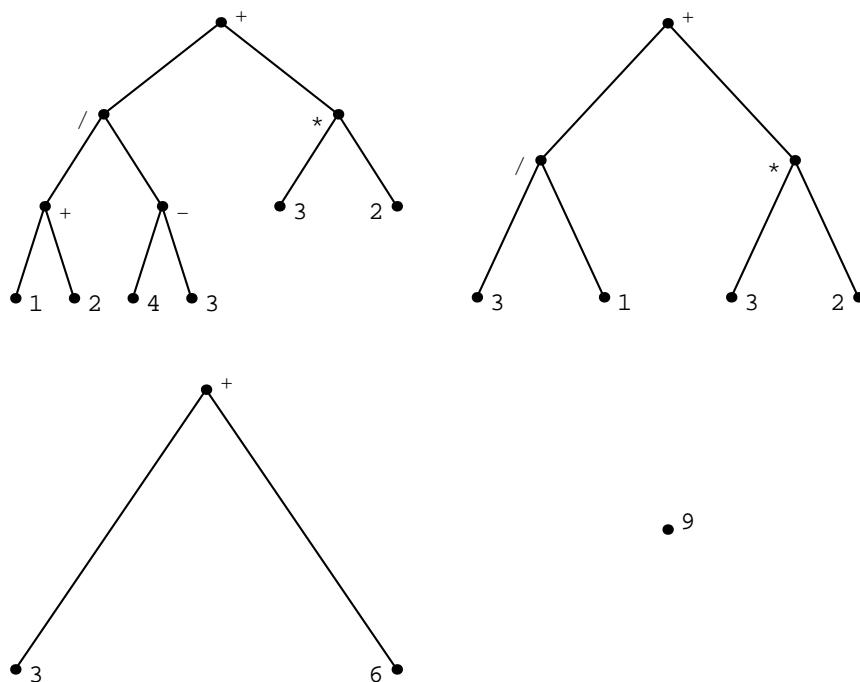
$$+/ + ab - xy \cdot 32$$

lub listy postfiksowej

$$ab + xy - / 32 \cdot +$$

Na cześć Łukasiewicza notację prefiksową nazywa się *notacją polską*, a postfiksową *odwrotną notacją polską*. Można łatwo przekonać się, że notacje te wyznaczają jednoznacznie stosowne drzewo. Jest tak dlatego, że znamy stopnie wierzchołków odpowiadających operacjom algebraicznym.

Popularne od lat siedemdziesiątych kalkulatory Hewletta-Packarda używają właśnie odwrotnej notacji polskiej do przeprowadzania rachunków. Kalkulatory te *celowo* nie posiadają nawiasów ani znaku równości. Mają nas bowiem przekonać i przyzwyczać do odwrotnej notacji polskiej. Zamiast tych przycisków kalkulator HP posiada klawisz ENTER, którym wprowadzamy dane na stos. Dla naszego wyrażenia z rys. 5.55 przykładowy rachunek na kalkulatorze HP w odwrotnej notacji polskiej wygląda następująco (po prawej stronie zaznaczamy stan stosu):



Rysunek 5.56: Kolejne etapy obliczania wyrażenia $(1+2)/(4-3)+3 \cdot 2$

1	1
ENTER	\square 1
2	2, 1
+	3
ENTER	\square 3
4	4, 3
ENTER	\square 4,3
3	3, 4, 3
-	1, 3
/	3
ENTER	\square 3
3	3, 3
ENTER	\square 3, 3
2	2, 3, 3
.	6, 3
+	9

Przyciśnięcie ENTER powoduje przesunięcie stosu o jedną pozycję w prawo, co powoduje powstanie wolnego miejsca na nową liczbę, \square . Natomiast przyciśnięcie klawisza operacji binarnej powoduje jej wykonanie na dwóch najbardziej po lewej stronie liczbach stosu, usunięcie tych liczb ze stosu oraz wpisanie wyniku w lewej części stosu. To działa!

5.18 Sieci zdarzeń

Teoria grafów znajduje też istotne zastosowanie w optymalizacji procesów technologicznych, produkcyjnych, w zarządzaniu itp. Mamy tam często do czynienia z tzw. *siecią zdarzeń*, czyli szeregiem procesów prowadzących do pewnego celu, jak wykonanie produktu. Niektóre z tych procesów są od siebie niezależne, a inne podlegają ograniczeniom typu *proces x może się rozpocząć po zakończeniu procesu y*. Opisana w tym podrozdziale technika nosi nazwę PERT (Programme Evaluation and Review Technique) [2,8]. Aby rozważyć przykład ilustrujący bardzo dobrze ten problem, przeistoczymy się na chwilę w cukiernika i rozważmy następujący przepis na szarlotkę:

Szarlotka Babuni

- a) masło posiekać z mąką (2 min)
- b) białyk ubić na pianę z cukrem i proszkiem do pieczenia (4 min)
- c) zagnieść ciasto (5 min)
- d) obrać, poszatkować, posłodzić jabłka (10 min)
- e) usmażyć jabłka na marmoladę (20 min)
- f) 2/3 ciasta rozwałkować, ułożyć spód w formie i piec w gorącym piekarniku (15 min)
- g) wlać masę jabłkową do formy (1 min)

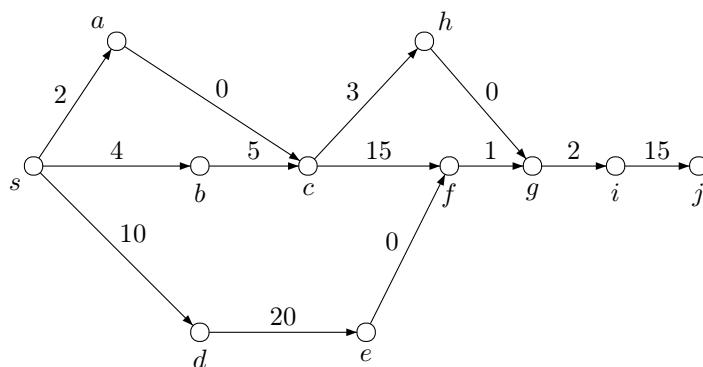
- h) z $1/3$ ciasta wyciąć paski (3 min)
- i) ułożyć paski w kratkę na górze masy (2 min)
- j) posmarować mlekiem i piec w gorącym piekarniku (15 min)

Już na pierwszy rzut oka jest oczywiste, że niektóre czynności przepisu mogą przebiegać równolegle, np. pieczenie spodu f) i przygotowanie masy jabłkowej d), e), podczas gdy inne mogą się zacząć dopiero po ukończeniu czynności poprzedzającej, np. aby piec ciasto, musimy je najpierw zagnieść itp. Powyższy przepis przedstawiony jest na rys. 5.57 jako sieć zdarzeń. Jest to graf skierowany z wagami odpowiadającymi czasowi wykonania danej czynności. Rozpoczynamy w punkcie startowym s w czasie 0. Możemy jednocześnie rozpoczęć czynności a), b) i d) – zauważmy, że jesteśmy cukiernikiem, a nie gospodynią domową, mamy więc do dyspozycji kuchcików, którzy mogą wykonywać odpowiednie czynności jednocześnie. Rysujemy więc trzy krawędzie wychodzące z s , a kończące się na wierzchołkach a , b i c , oznaczających zakończenie danej czynności. Wagi krawędzi oznaczają czas wykonania czynności wzięty z przepisu. Czynność c), czyli zagniatanie ciasta, możemy rozpoczęć po zakończeniu czynności a) i b). Aby to zaznaczyć, wprowadzamy pomocniczą krawędź idącą od a do c o wadze 0. Krawędź ta oznacza, że c) nie może się rozpoczęć przed zakończeniem a). W ogólności, czynność w) może się rozpoczęć dopiero po zakończeniu wszystkich czynności, od których krawędzie wchodzą do wierzchołka w . Dalsza konstrukcja grafu jest oczywista.

Interesują nas teraz następujące pytania:

1. Jak długo trwa cały proces wyrobu szarlotki?
2. Które czynności możemy wydłużyć bez opóźniania całego procesu?
3. Jak możemy zoptymalizować proces, tzn. które etapy trzeba skrócić, aby przyspieszyć produkcję, gdzie przydzielić więcej kuchcików itd.?

Najpierw zbadajmy, o jakim czasie mogą kończyć się poszczególne etapy przepisu. Zaczynamy w chwili 0. Czasy zakończenia czynności ukazane są jako etykiety wierzchołków na rys. 5.58. Dla każdego wierzchołka jest to największa waga ze

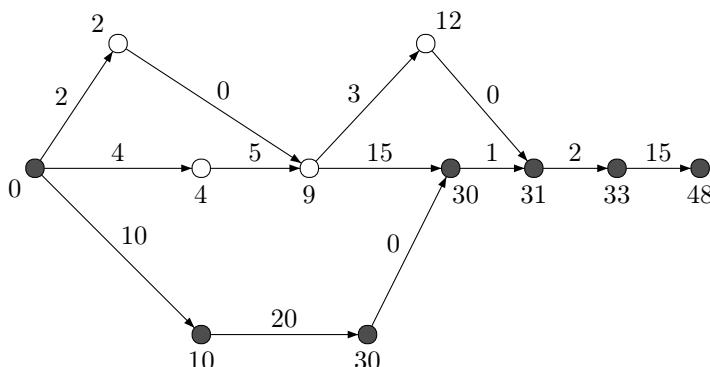


Rysunek 5.57: Przepis na szarlotkę jako sieć zdarzeń

wszystkich prowadzących doń dróg. Na przykład czas zakończenia czynności f wynosi 30 min, bo jest to waga drogi $sdef$, która jest większa od wagi drogi $sbcf$, wynoszącej tylko 24 min. Drogą o najdłuższym czasie biegącą od s do końcowego wierzchołka j jest $sdefgij$. Wierzchołki leżące wzdłuż tej drogi zostały wytluszczone. Taka droga o największej wadze (sumarycznym czasie wykonywania czynności) nazywa się *drogą krytyczną*. Jest oczywiste, że skrócenie czasu trwania czynności na drodze krytycznej przyspieszy cały proces. W naszym przypadku produkcja szarlotki trwa 48 min. Z drugiej strony, czynności poza drogą krytyczną, a), b), c) i h), mogą być wykonane nieco wolniej (zaraz zobaczymy, o ile) i nie wpłynie to na czas wykonywania całego procesu.

Definiujemy czas $T(v, w)$ jako maksymalną wagę drogi od v do w . Wprowadzamy też wielkość $K(w) = T(s, j) - T(w, j)$, gdzie j jest czynnością końcową. Jest to więc różnica czasu całego procesu oraz czasu, w którym można wykonać czynności od w do j , co prowadzi natychmiast do interpretacji $K(w)$ jako najpóźniejszego momentu, w którym musimy zakończyć czynność w w taki sposób, aby nie opóźnić całego procesu. Wreszcie określmy rezerwę czasową wierzchołka jako $R(w) = K(w) - T(s, w)$. Jest to maksymalny czas, o jaki możemy opóźnić wykonywanie procesów „dochodzących” do wierzchołka w .

Tabela 5.6 ukazuje te czasy dla rozważanego przykładu. Widzimy natychmiast, że wierzchołki na linii krytycznej mają rezerwę czasową zero, czyli nie można ich opóźnić bez opóźnienia całego procesu. Natomiast pozostałe wierzchołki mają większą od zera rezerwę czasową. Oznacza to, że możemy „bezkarne” opóźnić wykonanie procesu odpowiadającego danemu wierzchołkowi w o czas $R(w)$. Istotnie, $R(a) = 13$, więc czas siekania masła z mąką możemy wydłużyć do 15 min. – wtedy czas osiągnięcia wierzchołka f wzdłuż drogi $sacf$ wyniósłby 30 min., tyle samo co czas wzdłuż drogi $sdef$. Wówczas graf posiadałby dwie drogi krytyczne. Zauważmy jeszcze, że jeśli użyjemy całości lub części rezerwy czasowej dla jednego wierzchołka, musimy przeliczyć sieć, tzn. określić czasy K i R jeszcze raz, i dopiero wtedy możemy użyć rezerwy czasowej dla innego wierzchołka.



Rysunek 5.58: Sieć zdarzeń z podanymi czasami zakończenia kolejnych czynności $T(s, w)$

Tabela 5.6: Wielkości występujące w analizie PERT pieczenia szarlotki Babuni:

$T(s, w)$ – maksymalny czas od startu do ukończenia czynności w , $T(w, j)$ – czas od ukończenia czynności w do końca procesu, $K(w)$ – najpóźniejszy czas, w jakim można rozpocząć czynność w bez opóźniania całego procesu, $R(w)$ – czas, o jaki można opóźnić wykonywanie czynności w bez opóźniania całego procesu (rezerwa czasowa wierzchołka w)

	s	a	b	c	d	e	f	g	h	i	j
$T(s, w)$	0	2	4	9	10	30	30	31	12	33	48
$T(w, j)$	48	33	38	33	38	18	18	17	17	15	0
$K(w)$	0	15	10	15	10	30	30	31	31	33	48
$R(w)$	0	13	6	6	0	0	0	0	19	0	0

Algorytm znajdujący drogę krytyczną jest bardzo podobny do algorytmu najkrótszej ścieżki 5.6 z tą różnicą, że teraz szukamy najdłuższej drogi (drogi o największej wadze).

5.19 Przepływy w sieciach

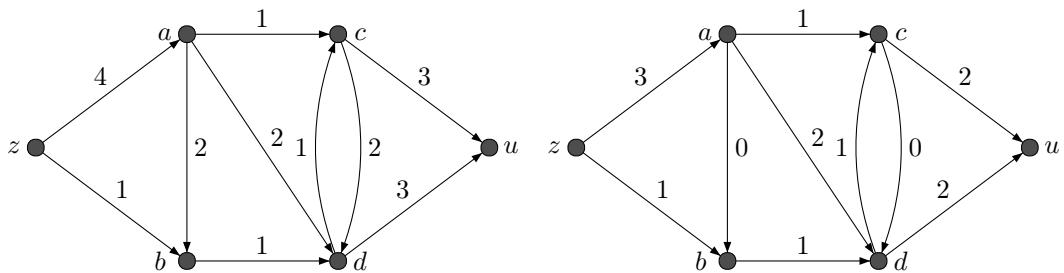
Kolejna bardzo ważna aplikacja teorii grafów to badanie *przepustowości sieci*. Z problemem tym spotykamy się na co dzień, od hydrauliki, poprzez komunikację drogową, do telekomunikacji, sieci informatycznych itd. Podstawowe pytanie o ogromnym znaczeniu praktycznym, to *jaka jest największa prędkość (przepustowość) przepływu wody, samochodów czy informacji między dwoma miejscami w sieci*. Na początek kilka definicji.

Def. 5.45. Siecią nazywamy graf skierowany, w którym każdej krawędzi m przyporządkowujemy nieujemną liczbę $\psi(m)$, zwaną przepustowością krawędzi.

Def. 5.46. Stopniem wejściowym wierzchołka v względem funkcji ψ nazywamy sumę przepustowości wszystkich wchodzących doń krawędzi m_v , tj. wielkość $s_\psi(v) = \sum_{m_v} \psi(m_v)$. Stopniem wyjściowym wierzchołka v względem funkcji ψ nazywamy sumę przepustowości wszystkich wychodzących zeń krawędzi m_v , tj. $t_\psi(v) = \sum_{m_v} \psi(m_v)$.

Def. 5.47. Źródłem z nazywamy wierzchołek o stopniu wejściowym $s(z) = 0$ i stopniu wyjściowym $t(z) > 0$, a ujściem u nazywamy wierzchołek o stopniu wyjściowym $t(u) = 0$ i stopniu wejściowym $s(u) > 0$.

Przykład sieci o jednym źródle i jednym ujściu pokazano na rys. 5.59. Okazuje się, że wystarczy rozważyć taki standardowy przypadek, ponieważ analizę sieci o kilku źródłach z_i i kilku ujściach u_j można sprowadzić do przypadku o jednym źródle i jednym ujściu. Możemy bowiem połączyć wszystkie źródła z_i z nowym pomocniczym źródłem Z w taki sposób, że przepustowość każdej



Rysunek 5.59: Przykład sieci z zaznaczonymi przepustowościami krawędzi ψ (lewa część) oraz z zaznaczonym przepływem φ (prawa część). Graf ma jedno źródło z i jedno ujście u

dodanej do z_i krawędzi równa się stopniowi wyjściowemu źródła z_i . Podobnie postępujemy dla ujść. Oczywiście w wyniku tej konstrukcji $t_\psi(Z) = \sum_i t_\psi(z_i)$ oraz $s_\psi(U) = \sum_j s_\psi(u_i)$. Przepustowości krawędzi grafu z rys. 5.59 oznaczono liczbami przy krawędziach. Większość połączeń między wierzchołkami jest jednokierunkowa, jedynie wierzchołki c i d połączone są dwukierunkowo.

Możemy sobie tutaj wyobrazić następujący problem komunikacyjny: *ile pojazdów może przejechać od punktu z do punktu u w ciągu godziny*, gdzie przepustowości krawędzi oznaczają ile maksymalnie pojazdów może przejechać daną drogą w ciągu godziny (np. w tys./godz.). Do dalszej analizy wprowadzimy pojęcie przepływu w sieci.

Def. 5.48. Przepływem w sieci nazywamy funkcję φ , która każdej krawędzi m przyporządkowuje liczbę rzeczywistą spełniającą warunki:

1. $0 \leq \varphi(m) \leq \psi(m)$.
2. Stopień wejściowy i wyjściowy względem funkcji φ wszystkich wierzchołków z wyjątkiem źródła i ujścia jest równy, $s_\varphi(v) = t_\varphi(v)$, $t \neq z, u$.

Warunek 1 oznacza, że przepływ wzduż danej krawędzi nie może przekroczyć jej przepustowości, natomiast warunek 2 oznacza, że to, co płynie, *nie jest magazynowane* w wierzchołkach: dokładnie tyle, ile w jednostce czasu wpływa do wierzchołka, musi zeń wypływać. Warunek ten nazywa się też warunkiem ciągłości, a w kontekście sieci elektrycznych dobrze nam znanym pierwszym prawem Kirchhoffa³⁹: *suma natężen prądów wpływających do węzła sieci równa się sumie natężen prądów zeń wypływających*. Wyjątkami są tu źródło i ujście. Oczywiście, prawo ciągłości jako prawo przyrody jest spełnione powszechnie, a jego niespełnienie w źródle i ujściu wynika jedynie z odrębnego traktowania tych wierzchołków: prąd do źródła dociera w inny sposób, „inną, nienarysowaną krawędzią” niż w obrębie rozważanej przez nas sieci, podobnie odbieramy prąd z ujścia poprzez „krawędź spoza naszej sieci”.

Przykładowy przepływ pokazany jest po prawej stronie rys. 5.59. Krawędzie, dla których przepływ jest równy przepustowości, $\varphi(m) = \psi(m)$, nazywamy

³⁹ Gustav Robert Kirchhoff (1824-1887), niemiecki fizyk.

krawędziami *nasyconymi*. W naszym przykładzie krawędzie (z, b) , (a, c) , (a, d) , (b, d) oraz (d, c) są nasycone.

Tw. 5.30. *Suma stopni wejściowych względem przepływu wszystkich wierzchołków sieci jest równa sumie stopni wyjściowych wszystkich wierzchołków sieci, $\sum_v t_\varphi(v) = \sum_v s_\varphi(v)$.*

Dowód: Twierdzenie jest uogólnieniem lematu o uściskach dłoni. Dana krawędź m wychodząca z wierzchołka v i wchodząca do w wnosi do $t_\varphi(v)$ i $s_\varphi(w)$ jednakowe przyczynki, mianowicie $\varphi(m)$. Usunięcie tej krawędzi nie zmienia więc wartości różnicę $\sum_v (t_\varphi(v) - s_\varphi(v))$. Usuwając po kolejne wszystkie krawędzie, dochodzimy do grafu pustego, dla którego suma stopni wyjściowych i wejściowych wynosi zero. Zatem $\sum_v (t_\varphi(v) - s_\varphi(v)) = 0$. \square

Wniosek 5.5. *Ponieważ wierzchołki niebędące źródłem ani ujściem mają równe stopnie wejściowe i wyjściowe, stopień wyjściowy źródła równa się stopniowi wejściowemu ujścia, $t_\varphi(z) = s_\varphi(u) = p$. Liczbę p nazywamy wielkością przepływu.*

Wniosek ten oznacza po prostu, że to, co wpływa do sieci, wypływa z niej ujściem, zgodnie z prawem ciągłości lub popularną maksymą „nic w Przyrodzie nie ginie”.

To co nas interesuje z punktu widzenia optymalizacji przepływu, to *przepływ maksymalny*:

Def. 5.49. *Przepływem maksymalnym nazywamy przepływ o największej możliwej wielkości przepływu p .*

Możliwa jest sytuacja, w której dana sieć ma więcej niż jeden przepływ maksymalny. Rzut oka na rys. 5.59 pokazuje, że ukazany tam przepływ jest maksymalny. Istotnie, nie możemy go zwiększyć, bo krawędzie (a, c) , (a, d) i (b, d) są wysycone. Inny przepływ maksymalny w tej sieci ma $\varphi'((d, c)) = 0$, $\varphi'((c, u)) = 1$, $\varphi'((d, u)) = 3$, a przepływy wzduż pozostałych krawędzi są takie same jak na prawej części rys. 5.59.

Aby podać podstawowe twierdzenie dotyczące przepływów w sieciach, zdefiniujemy pojęcie *przekroju*.

Def. 5.50. *Przekrojem sieci nazywamy podział zbioru wierzchołków V na zbiór S i $T = V \setminus S$, taki, że $z \in S$ oraz $u \in T$. Jest więc $2^{|V|-2}$ różnych przekrojów. Przepustowość przekroju nazywamy sumę przepustowości wszystkich krawędzi m o początku w wierzchołku $v \in S$ i końcu w wierzchołku $w \in T$, tzn. wielkość $P(S) = \sum_{m:v \in S, w \in T} \varphi(m)$.*

Innymi słowy, przekrój rozspaja sieć na dwie niepołączone części, przy czym źródło należy do jednej z nich, a ujście do drugiej. Przykładami przekrojów dla

sieci z rys. 5.59 są podziały

$$\begin{aligned} S_1 &= \{z\}, \quad T_1 = \{a, b, c, d, u\}, \\ P(S_1) &= \psi((z, a)) + \psi((z, b)) = 4 + 1 = 5, \\ S_2 &= \{z, a\}, \quad T_2 = \{b, c, d, u\}, \\ P(S_2) &= \psi((z, b)) + \psi((a, b)) + \psi((a, c)) + \psi((a, d)) = 6, \\ S_3 &= \{z, a, b\}, \quad T_3 = \{c, d, u\}, \\ P(S_3) &= \psi((b, d)) + \psi((a, c)) + \psi((a, d)) = 4, \text{ itd.} \end{aligned}$$

Def. 5.51. Przekrój minimalny to przekrój o najmniejszej przepustowości, tj. o najmniejszej wartości $P(S)$.

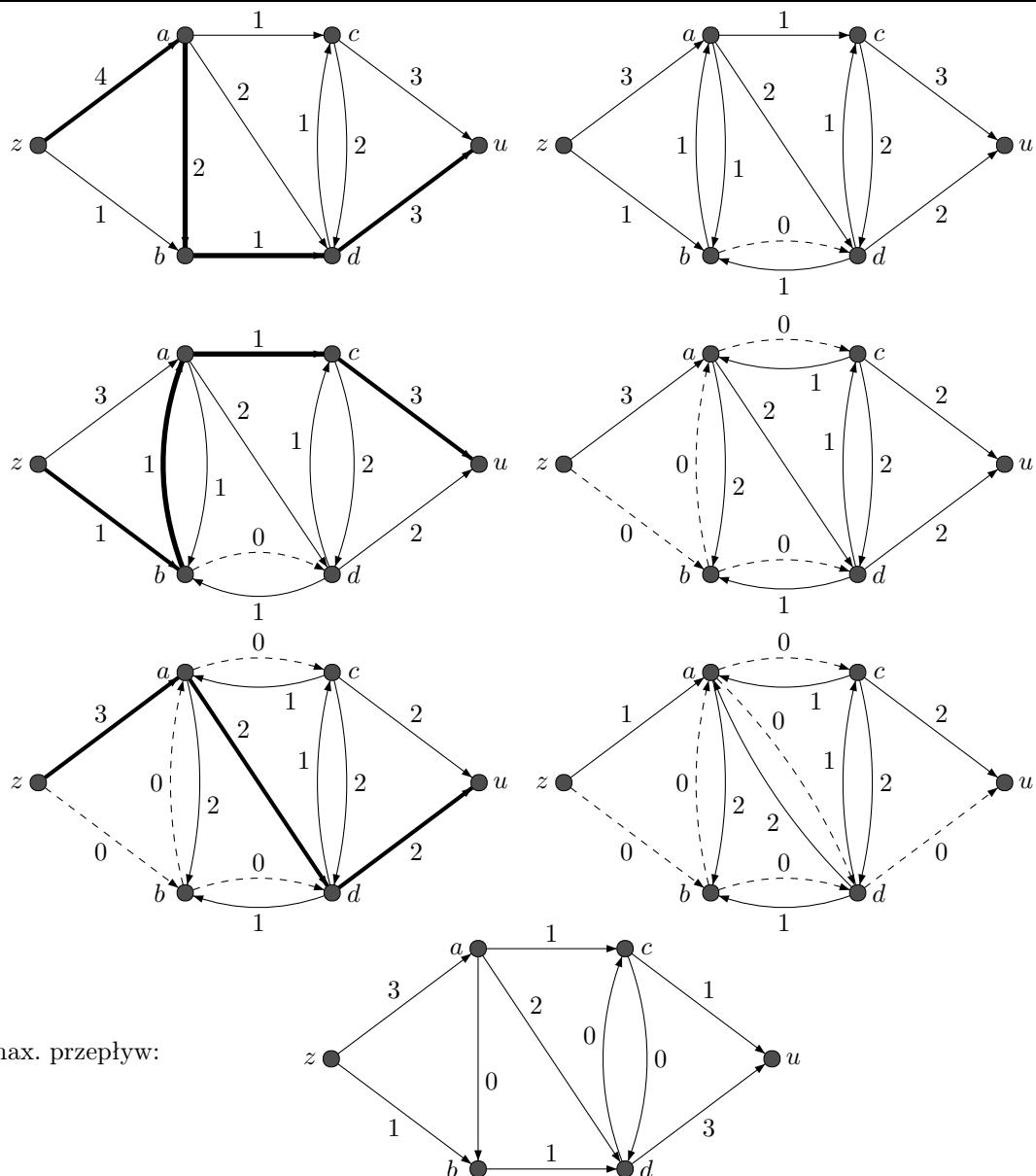
Przekrojem minimalnym dla sieci z rys. 5.59 jest przekrój S_3 o przepustowości $P(S_3) = 4$. A teraz podstawowe twierdzenie dotyczące przepływu w sieciach:

Tw. 5.31 (o maksymalnym przepływie i minimalnym przekroju). W dowolnej sieci przepływ maksymalny jest równy przepustowości minimalnego przekroju.

Jest oczywiste, że wartość przepływu nie może przekroczyć przepustowości dowolnego przekroju. Dowód twierdzenia w drugą stronę, mianowicie że maksymalny przepływ wysyca przepustowość minimalnego przekroju, można znaleźć np. w [8].

Podstawowym narzędziem znajdującym maksymalny przepływ w danej sieci wyjściowej o zbiorze wierzchołków V , zbiorze krawędzi E i przepustowości ψ jest algorytm Forda-Fulkersona⁴⁰ i jego modyfikacje. Obiektem roboczym jest tu pomocnicza sieć rezydualna o zbiorze wierzchołków $V_R = V$, tym samym źródle i ujściu, zbiorze krawędzi E_R i funkcji przepustowości ψ_R , w której w kolejnych krokach algorytmu przepustowość jest odpowiednio modyfikowana przez „wpompowywany” przepływy. Mówiąc w uproszczeniu, krawędzie sieci rezydualnej mają przepustowości pozostałe po ustaleniu w sieci pewnego przepływu, który „jest odejmowany” (patrz poniżej) od przepustowości krawędzi sieci wyjściowej. W rezultacie przepustowość rezydualna pozostaje do ewentualnego wykorzystania przez dodatkowy przepływ. Należy podkreślić, że sieć rezydualna jest zbudowana na tych samych wierzchołkach V co sieć wyjściowa, ale w ogólności może mieć inne krawędzie E_R niż E , gdyż w trakcie realizacji algorytmu z E_R usuwane są krawędzie o zerowej przepustowości rezydualnej oraz powstają dodatkowe krawędzie idące w przeciwnym kierunku niż krawędzie sieci wyjściowej. Jest to bardzo sprytna sztuczka, pozwalająca de facto na zmniejszenie przepływu w danej krawędzi poprzez odwrócenie kierunku dodawanego przepływu.

⁴⁰ Delbert Ray Fulkerson (1924-1976), amerykański matematyk.



Rysunek 5.60: Kolejne kroki działania algorytmu Forda-Fulkersona dla przykładowej sieci z rys. 5.59 (3 górne wiersze) i uzyskany maksymalny przepływ (dolny wiersz)

Alg. 5.16 (algorytm Forda-Fulkersona).

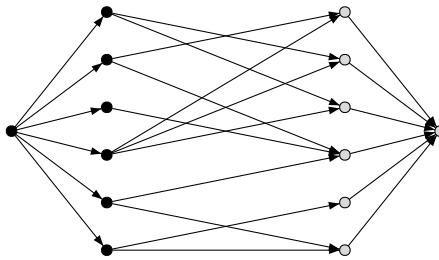
1. Określ sieć rezydualną jako równą sieci wyjściowej, o tym samym zbiorze wierzchołków, ujściu iźródle. Zainicjalizuj zbiór krawędzi jako $E_R = E$ o przepustowościach $\psi_R(v, w) = \psi(v, w)$. Przydziel każdej krawędzi (v, w) przepływu $\varphi(v, w) = 0$.

2. Jeżeli w sieci rezydualnej nie ma ścieżki od z do u STOP. Maksymalny przepływ dla sieci wyjściowej oblicz (dla (u, v) należących do E) jako różnicę $\varphi(u, v) = \psi(u, v) - \psi_R(u, v)$.
3. (wszystkie operacje na sieci rezydualnej) Znajdź dowolną ścieżkę s prowadzącą od z do u oraz minimalną przepustowość p wśród jej krawędzi. Dla każdej krawędzi (v, w) ścieżki s :
 - Zmniejsz przepustowość (v, w) o p .
 - Jeśli przepustowość (v, w) wynosi 0, usuń krawędź z grafu.
 - Jeśli sieć nie zawiera krawędzi (w, v) (tj. idącej w odwrotnym kierunku do (v, w)), dodaj ją i przydziel przepustowość 0. Operacji tej nie musisz wykonywać dla $u = z$ lub $v = u$.
 - Zwiększ przepustowość (w, v) o p .

4. Idź do kroku 2.

Kolejne kroki algorytmu przedstawione są na rys. 5.60 dla przykładowej sieci z rys. 5.59. Zaczynamy od sieci rezydualnej równej sieci wyjściowej i znajdujemy ścieżkę $s = zabdu$, oznaczoną wytłuszczeniem na lewym górnym rysunku 5.60. Najmniejsza przepustowość p wśród krawędzi tej ścieżki wynosi 1, a więc o tyle zmniejszamy przepustowości krawędzi wzdłuż s . Krawędź (b, d) ma teraz przepustowość 0, więc ją usuwamy (usunięte krawędzie oznaczamy liniami przerywanymi). Dodajemy krawędzie idące w odwrotną stronę, (b, a) i (d, b) , przydzielamy im przepustowości 0, po czym zwiększamy je o $p = 1$. Efekt tych operacji prowadzi do sieci rezydualnej z prawego górnego rysunku 5.60. W sieci tej możemy wybrać kolejną ścieżkę od z do u , np. $zbacu$ (lewy rysunek drugiego wiersza), dla której $p = 1$. Zauważmy, że wykorzystujemy tu dodaną uprzednio krawędź (b, a) , co jest równoważne zmniejszeniu przepływu wzdłuż krawędzi (a, b) . Pomniejszamy przepustowości krawędzi wzdłuż ścieżki o 1, usuwamy krawędzie o zerowej przepustowości oraz dodajemy krawędź (c, a) , która uzyskuje przepustowość 1. Wynik to prawy rysunek drugiego wiersza. Trzeci wiersz ukazuje kolejną iterację, tym razem ze ścieżką $zadu$ o $p = 2$. Po jej wykonaniu nie ma już więcej ścieżek od z do u , przechodzimy więc do obliczenia maksymalnego przepływu. Od przepustowości krawędzi zaznaczonych na lewym rysunku pierwszego wiersza odejmujemy przepustowości z prawego rysunku trzeciego wiersza. Wynik ukazany jest w ostatnim rzędzie rys. 5.60. Zauważmy jeszcze, że suma znalezionych p dla wszystkich ścieżek daje wartość maksymalnego przepływu, w tym przypadku 4.

Ciekawostką jest fakt, że algorytm 5.16 nie zawsze się zatrzymuje i może wykonywać się w nieskończoność! Dzieje się tak w przypadku pewnych sieci z odpowiednio dobranymi rzeczywistymi przepustliwościami i dla pewnej sekwencji wyboru ścieżek (problem nie pojawia się dla przepustowości będących liczbami naturalnymi). Algorytm może wówczas dążyć do rozwiązania, które



Rysunek 5.61: Sieć służąca do rozwiązywania problemu maksymalnego skojarzenia w grafie z rys. 5.50

nie jest maksymalnym przepływem (zob. np. http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm). Modyfikacją algorytmu Forda-Fulkersona wolną od tej patologii jest algorytm Edmondsa⁴¹-Karpa⁴², uzupełniający procedurę z 5.16 o ścisły przepis znajdowania ścieżki, mianowicie za pomocą przeszukiwania wszerz. Można pokazać, że algorytm ten zawsze się kończy i ma złożoność obliczeniową $\mathcal{O}(VE^2)$. Algorytm Dinica⁴³ o złożoności $\mathcal{O}(V^2E)$ jest jeszcze szybszy.

Na koniec podrozdziału zauważmy, że algorytm znajdujący maksymalny przepływ w sieci pozwala również na rozwiązywanie problemu kojarzenia małżeństw z podrozdziału 5.15! W tym celu przykładowy graf z rys. 5.50 zmieniamy w sieć ukazaną na rys. 5.61 poprzez dodanie źródła połączonego z wszystkimi kobietami i ujścia połączonego z wszystkimi mężczyznami. Przepustowość każdej krawędzi wynosi 1. Z konstrukcji wynika, że znalezienie maksymalnego przepływu o przepustowości n jest równoważne znalezieniu maksymalnego skojarzenia w wyjściowym grafie, co możemy uzyskać w czasie wielomianowym. W dość nie-spodziewany sposób hydraulika wiąże się ze swataniem!

Ćwiczenia

- 5.1. Wykonaj algorytm 5.1 na grafie z rys. 5.9.
- 5.2. Jaką interpretację ma m -ta potęga macierzy sąsiedztwa, S^m , dla grafu prostego (nieskierowanego lub skierowanego)?
- 5.3. Na podstawie poprzedniego zadania wykaż, że dla grafu prostego nieskierowanego ślad macierzy S^2 równa się podwojonej liczbie krawędzi, a ślad S^3 równa się sześciokrotnej liczbie trójkątów zawartych w tym grafie (ślad macierzy A to suma jej elementów diagonalnych, $\text{Tr } A = \sum_i A_{ii}$).
- 5.4. Ile trójkątów zawartych jest w grafie K_n ?
- 5.5. Zadanie o wilku, kozie i kapuście. Przewoźnik musi przeprowadzić promem na drugą stronę rzeki wilka, kozę i kapustę. Łódź jest na tyle mała, że może

⁴¹ Jack R. Edmonds (1934-), kanadyjski matematyk i informatyk.

⁴² Richard Manning Karp (1935-), wybitny amerykański informatyk.

⁴³ Efim Dinic, rosyjsko-izraelski informatyk.

pomieścić oprócz przewoźnika tylko jeden „obiekt”. Jak ma to zrobić, aby móc przypilnować, by wilk nie zjadł kozy, a koza kapusty? Narysuj graf ilustrujący możliwe sytuacje i zaznacz rozwiązanie. Ile najmniej przepraw potrzeba? Ile jest takich optymalnych rozwiązań?

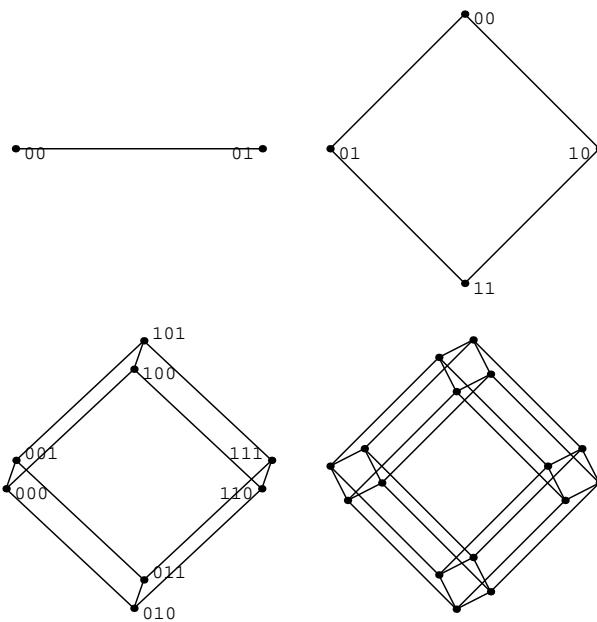
- 5.6. Jaki jest najmniejszy wymiar kwadratowej szachownicy, dla którego istnieje rozwiązanie problemu konika szachowego z podrozdziału 5.2? Zagadnieniem tym zajmował się już Euler!
- 5.7. Narysuj wszystkie nieizomorficzne grafy proste 3-regularne o sześciu wierzchołkach (graf k -regularny to graf prosty o stopniu każdego wierzchołka równym k). Dla dowolnego z narysowanych grafów wyznacz macierz incydencji, macierz sąsiedztwa oraz listy incydencji.
- 5.8. Cząsteczkę węglowodoru o wzorze sumarycznym C_kH_{2k+2} można przedstać w postaci spójnego grafu prostego, w którym wierzchołki oznaczają atomy węgla (C) lub wodoru (H), a krawędzie wiązania chemiczne. Każdy atom wodoru związany jest z jednym innym atomem, a każdy atom węgla z czterema innymi atomami. Narysuj kilka grafów dla małych wartości k . Pokaż, że każdy graf reprezentujący C_kH_{2k+2} jest drzewem. Nieizomorficzne grafy dla ustalonego k nazywamy izomerami. Ile jest izomerów dla $k = 1, 2, 3, 4, 5$?
- 5.9. Rozważ zbiór wszystkich grafów o n wierzchołkach. Graf jest wybierany losowo z tego zbioru. Jakie jest prawdopodobieństwo, że graf
 - (a) jest grafem cyklicznym o długości n ?
 - (b) zawiera trójkąt?
- 5.10. Do trzech budowanych domów należy doprowadzić ziemią linie zasilania prądem, wodą i gazem. Czy można to zrobić bez krzyżowania tych linii?
- 5.11. Na balu jest n mężczyzn i $m \geq n$ kobiet. Ile najmniej tańców potrzeba, aby każda kobieta zatańczyła z każdym partnerem?
- 5.12. Znajdź wielościany foremne na torusie.
- 5.13. Znajdź algorytm dla kilku pierwszych wartości liczby krążków n dla łamigłówki Reve'a (uogólnienie wież Hanoi dla czterech prętów).
- 5.14. n -wymiarowa hiper kostka Q_n zdefiniowana jest jako następujący graf: zbiór wierzchołków to wszystkie n -wyrazowe ciągi liczb 0 i 1, np. dla $n = 2$ zbiór wierzchołków to zbiór punktów na płaszczyźnie

$$\{(0, 0), (0, 1), (1, 0), (1, 1)\},$$

a dla $n = 3$

$$\{(0, 0, 0), (0, 0, 1), (0, 1, 0), (1, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$$

itd. dla większej liczby wymiarów n (zob. rys. 5.62). Z definicji, wierzchołki są sąsiednie, jeśli określające je ciągi współrzędnych różnią się w dokładnie jednym miejscu, np. dla $n = 3$ wierzchołki $(1, 0, 1)$ i $(1, 0, 0)$ są sąsiednie,

Rysunek 5.62: Hiperkostki Q_n dla $n = 1, 2, 3, 4$

a $(1, 0, 1)$ i $(1, 1, 0)$ nie są. Udowodnij, że liczba wierzchołków Q_n wynosi 2^n , a liczba krawędzi $n2^{n-1}$. Pokaż, że Q_n jest grafem dwudzielnym.

- 5.15. Narysuj wszystkie nieizomorficzne hamiltonowskie grafy proste o pięciu wierzchołkach.
- 5.16. Narysuj wszystkie nieizomorficzne eulerowskie grafy proste o pięciu wierzchołkach.
- 5.17. Problem chińskiego listonosza (zaproponowany przez matematyka chińskiego Mei Ku Kwana w 1962 r.) polega na tym, aby w grafie o krawędziach z wagami równymi ich długości znaleźć najkrótszy cykl przechodzący przez każdą krawędź co najmniej raz. Rozwiąż ten problem.
- 5.18. Niech „liczbą sąsiednią” dla danej liczby w trójkącie Pascala będzie liczba sąsiadująca w tym samym wierszu lub liczba w wierszu powyżej nad daną liczbą nieco w lewo lub w prawo (tak jak bierzemy dla rekurencji). Narysuj duży trójkąt Pascala i połącz w nim sąsiednie liczby *nieparzyste*. Co otrzymałeś?
- 5.19. Zaprojektuj komputer analogowy znajdujący minimalne drzewo Steinera dla pola naftowego w terenie górzystym.
- 5.20. Pokaż, że mapa z rys. 5.42 nie jest trójbarwna.
- 5.21. Ile najmniej kolorów potrzeba do pomalowania ścian (a) graniastosłupa (b) ostrosłupa (c) brył platońskich z rys. 5.14, aby sąsiednie ściany miały różny kolor?

- 5.22. Pokaż przez wskazanie przykładu, że graf K_5 pokolorowany krawędziowo dwoma kolorami może nie zawierać kliki K_3 o krawędziach tego samego koloru.
- 5.23. Znajdź liczbę Ramseya $R(n, 2)$.
- 5.24. Na gęsto zalesionym terytorium Kanady i powierzchni 10000 km^2 wydzielono 100 działek łowieckich o nieregularnym kształcie, każdą o powierzchni 100 km^2 . Jednocześnie strażacy podzieliли ten teren na 100 obszarów podlegających ochronie przeciwpożarowej niepokrywających się z działkami łowieckimi, każdy również o powierzchni 100 km^2 . Czy jest możliwe rozmieszczenie 100 wież obserwacyjnych w taki sposób, aby i myśliwi, i strażacy mogli wspólnie użytkować po jednej wieży na każdym ze swoich obszarów?
- 5.25. Z potasowanej tali 52 kart widz wyciąga w dowolny sposób 5 kart nie pokazując ich iluzjonistce. Następnie przekazuje je jawnie asystentowi, który wybiera spośród nich 4 karty i pokazuje je po kolej iluzjonistce. „Aha, piątą kartą jest...” – mówi iluzjonista, bezbłędnie odgadując. Jak to możliwe?

Rozdział 6

Algorytmy

W tym wykładzie mieliśmy już wiele razy do czynienia z różnymi algorytmami. Intuicyjnie wiedzieliśmy, o co chodzi: algorytm to zestaw instrukcji, pewien dobrze określony przepis, który wykonujemy na jakichś danych wejściowych (które możemy zmieniać) i po ukończeniu (a algorytm powinien posiadać tę miłą cechę, że się kończy) otrzymujemy wynik. Każdy rachunek, który wykonujemy, każda obróbka danych, każdy program komputerowy, jest pewnym algorytmem.

6.1 Czym jest algorytm

Pojecie algorytmu jest niemal tak stare, jak sama matematyka. Nazwa pochodzi od łacińskiej formy nazwiska słynnego matematyka perskiego (niektóre źródła określają go jako matematyka arabskiego pochodzenia uzbeckiego) al-Chwarizmiego¹. Od zarania algorytm był przepisem rozwiązywania konkretnych problemów matematycznych. W obecnej erze informatycznej niezbędne jest

¹ Muhammad ibn Musa al-Chwarizmi (780 – ok. 850), arabski matematyk, astronom, geograf, kartograf i astrolog, autor m.in. traktatu o rozwiązywaniu równań liniowych i kwadratowych, jeden z ojców algebry. Spopularyzował hinduski system zapisu liczb na Bliskim Wschodzie i w Europie.

bardziej precyzyjne zdefiniowanie, czym jest algorytm, co uczynimy w tym rozdziale. Co więcej, sam algorytm stał się obiektem formalnych badań matematycznych: możemy klasyfikować algorytmy, określać ich złożoność, badać, czy ich działanie się zakończy itd.

Każdy algorytm posiada następujące własności:

- Zestaw ściśle zdefiniowanych instrukcji możliwych do wykonania w każdej zaistniałej sytuacji, tj. dla wszystkich dopuszczalnych danych wejściowych.
- Dane wejściowe (mogą być puste).
- Dane wyjściowe (nie mogą być puste, inaczej nigdy byśmy się nie dowiedzieli, co algorytm wyliczył).
- Skończony czas operacji. Warunek ten implikuje skończoność używanej pamięci, bo gdyby użyta pamięć była niekończona, to potrzeba by nieskończonego czasu, aby ją zapisać.

Niektórzy autorzy dopuszczają jednak algorytmy, których działanie nie kończy się nigdy. Jeśli taki algorytm podczas wykonywania generuje dane wyjściowe, to mamy „na bieżąco” pewną informację o badanym obiekcie. Wyobraźmy sobie np. algorytm badający częstotliwość występowania cyfry 1 w rozwinięciu dziesiętnym liczby π . Wyliczając coraz to dalsze cyfry, algorytm wyświetla na ekranie komputera stosunek liczby znalezionych jedynek do liczby wszystkich zbadanych do danej chwili cyfr. Rozwinięcie liczby π jest nieskończone, więc algorytm nigdy się nie skończy, a jednak otrzymujemy użyteczną informację (a zawsze możemy nacisnąć CTRL-ALT-DEL jeśli bieżąca odpowiedź już nas satysfakcjonuje). Bardziej formalnie i precyzyjnie algorytm można zdefiniować za pomocą *maszyny Turinga*², którą omawiamy w rozdz. 6.3.

Algorytmy można klasyfikować ze względu na różne kryteria: złożoność (omówioną w podrozdziale 6.6), budowę czy metodę działania. Oto kilka typowych cech:

- *Iteracja*, czyli wielokrotne wykonywanie w pętli zestawu instrukcji, lub *rekurencja*, czyli wywoływanie przez algorytm samego siebie. Iteracja lub rekurencja są „duszą” prawie wszystkich algorytmów.
- *Szeregowość* (wszystkie kroki muszą być wykonywane sekwencyjnie, bo każdy następny zależy od wyniku poprzedniego) lub *równoległość* (niektóre operacje mogą być wykonywane jednocześnie np. na wieloprocesorowym komputerze).
- Algorytm może być *deterministyczny* lub *probabilistyczny*. Algorytmy deterministyczne mają wszystkie kroki dokładnie określone, podczas gdy algorytmy probabilistyczne „rzucają kośćmi”, tzn. dokonują losowań, aby wybrać w sposób przypadkowy dalszą drogę postępowania. Algorytmy probabilistyczne są bardzo użyteczne w atakowaniu w przybliżony sposób problemów trudnych, jak np. problem komiwojażera z rozdz. 6.5.

² Alan Mathison Turing (1912-1954), brytyjski matematyk, logik i kryptolog, jeden z ojców współczesnej informatyki.

- Algorytmy mogą dawać odpowiedzi dokładne (np. odpowiedź na pytanie, czy liczba 7212 jest podzielna przez 3) lub przybliżone (np. jaka jest wartość całki $\int_0^\infty dx \exp(-x^2)/[1 + \exp(-x)]$).

6.2 Automaty skończone

Najprostszym typem algorytmu jest tzw. *deterministyczny automat skończony*. Oto jego abstrakcyjna definicja:

Def. 6.1. *Skończonym modelem deterministycznym typu akceptującego nazywamy następującą piątkę: (A, S, s_0, K, p) , gdzie*

- A jest alfabetem czytanym przez automat,
- S jest skończonym zbiorem stanów automatu,
- $s_0 \in S$ jest stanem początkowym,
- K jest zbiorem stanów akceptujących,
- $p : A \times S \rightarrow S$ jest funkcją przejścia, przeprowadzającą automat ze stanu s w nowy stan s' , w zależności od przeczytanego znaku alfabetu.

Automat początkowo znajduje się w stanie s_0 . Następnie czyta po kolejni znaki słowa, przechodząc od stanu do stanu, aż do wyczerpania całego słowa. Znaki należą do alfabetu A . Jeśli po zakończeniu pracy automat znajduje się w którymś ze stanów akceptujących, to mówimy, że słowo jest akceptowane, czyli należy do *języka rozpoznawalnego* przez dany automat. Inaczej mówiąc, automat „sortuje” słowa na te, które mają jakąś cechę (należą do języka), oraz te, które doń nie należą. Akceptuje je lub nie, stąd nazwa *akceptujący* lub *rozpoznający*. Nazwa *deterministyczny* podkreśla fakt, że przy danym przeczytanym znaku wejściowym automat przechodzi jednoznacznie od aktualnego stanu do nowego stanu. Mówiąc inaczej, procedura jest powtarzalna: te same dane prowadzą zawsze do identycznego wykonania algorytmu i tego samego wyniku. Nazwa *skończony* oznacza, że liczba stanów jest skończona (choć może być bardzo duża).

Jako prosty przykład skończonego automatu deterministycznego podamy algorytm określający, czy dana liczba w zapisie dwójkowym jest podzielna przez 3. Oznaczmy cyfry tej liczby przez a_k , gdzie $a_k = 0$ lub 1. Tak więc w tym przypadku alfabet to zbiór $A = \{0, 1\}$. Liczba ma zapis $L = (a_n a_{n-1} \dots a_3 a_2 a_1 a_0)_2$ (indeks 2 oznacza, że zapis jest binarny), a zatem $L = a_0 + 2(a_1 + 2(a_2 + (\dots + 2a_n) \dots))$. Zadaniem automatu jest sprawdzić, czy dzielenie przez 3 daje resztę 0. Można to zrobić w sprytny sposób. Niech s_i oznacza stan, że aktualna reszta z dzielenia przeczytanej dotąd części liczby wynosi i . Przy dzieleniu przez 3 mamy trzy takie stany: s_0 (podzielność), s_1 (reszta 1) i s_2 (reszta 2). Ustawiamy automat w stanie s_0 (nie zgromadziliśmy jeszcze żadnej reszty) i po kolejni czytamy cyfry liczby, począwszy od najbardziej znaczącej a_n . Jeśli a_n wynosi 0, to pozostajemy w stanie s_0 (nie ma reszty z dzielenia), a jeśli wynosi 1, to przechodzimy

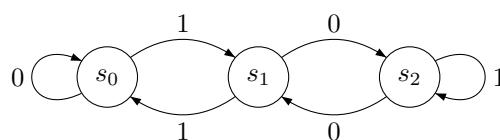
do stanu s_1 (reszta wynosi 1). Teraz czytamy kolejną cyfrę, a_{n-1} . Jeśli jesteśmy w stanie s_0 , to postępujemy jak przed chwilą, natomiast jeśli jesteśmy w stanie s_1 i przeczytaliśmy cyfrę 0, to dwie pierwsze cyfry naszej liczby wynoszą teraz 10. Jest to dwójkowy zapis liczby 2, więc zmieniamy stan automatu na s_2 . Jeśli jesteśmy w stanie s_1 i przeczytaliśmy cyfrę 1, to dwie pierwsze cyfry naszej liczby wynoszą teraz 11, czyli 3. Ale ponieważ interesuje nas podzielność, możemy prowadzić arytmetykę modulo 3, więc $3 = 0 \bmod 3$ i przechodzimy do stanu s_0 . W kolejnym kroku możemy być w stanie s_2 . Czytamy kolejny bit. Jeśli jest to 0, to otrzymujemy ciąg cyfr 100, czyli 4, a $4 = 1 \bmod 3$, zatem wracamy do stanu s_1 . Natomiast jeśli kolejna cyfra jest równa 1, to mamy ciąg 101, czyli $5 = 2 \bmod 3$, a więc pozostajemy w stanie s_2 . Reguły przejścia między stanami są zatem następujące:

$$\begin{aligned} s_0 &\xrightarrow{0} s_0, & s_0 &\xrightarrow{1} s_1, \\ s_1 &\xrightarrow{0} s_2, & s_1 &\xrightarrow{1} s_0, \\ s_2 &\xrightarrow{0} s_1, & s_2 &\xrightarrow{1} s_2. \end{aligned}$$

W powyższej notacji liczba nad strzałką oznacza przeczytaną cyfrę. Równoważny i bardziej konwencjonalny zapis to $p(s_0, 0) = s_0$, $p(s_0, 1) = s_1$ itd., pokazujący jawnie konstrukcję funkcji przejścia p .

Bardzo wygodne jest też zobrazowanie automatu w postaci graficznej, jak na rys. 6.1. Stany reprezentowane są przez wierzchołki grafu, a etykiety krawędzi pokazują, którą krawędzią należy pójść w przypadku przeczytania danej cyfry. A teraz przykład: rozważmy liczbę $(1001)_2$. Ciąg stanów automatu (najwygodniej posłużyć się rys. 6.1) to $(s_0, s_1, s_2, s_1, s_0)$. Po przeczytaniu całej liczby skończyliśmy w stanie s_0 , zatem liczba jest podzielna przez 3. Istotnie, w notacji dziesiętnej wyjściowa liczba to 9. A więc algorytm działa! Jedynym stanem akceptującym naszego automatu jest stan s_0 , zatem $K = \{s_0\}$. „Słowa” akceptowane to liczby w zapisie binarnym podzielne przez 3. Pozostałe słowa są odrzucane. Zbiór wszystkich akceptowanych słów stanowi *język rozpoznawalny* przez dany automat.

Weźmy inny przykład. Przepuśćmy przez naszą maszynkę liczbę $(11001)_2$. Automat prowadzi nas do stanu s_1 . Wnioskujemy stąd, że reszta z dzielenia liczby przez 3 wynosi 1. Istotnie, wyjściowa liczba w zapisie dziesiętnym to 25. Podobnie, ukończenie algorytmu w stanie s_2 oznaczałoby, że reszta z dzielenia



Rysunek 6.1: Skończony automat deterministyczny określający, czy dana liczba w zapisie dwójkowym jest podzielna przez 3

wejściowej liczby przez 3 wynosi 2. Widzimy więc, że nasz automat może spełniać bardziej użyteczną funkcję, niż stwierdzać podzielność, mianowicie może podawać resztę z dzielenia, co jest pełniejszą informacją. W tym celu określmy funkcję wyjściową $w : S \rightarrow B$, gdzie $B = \{0, 1, 2\}$ jest zbiorem możliwych reszt (B nazywamy alfabetem wyjściowym, w odróżnieniu od alfabetu wejściowego A). Mamy więc

$$w(s_i) = i, \quad i = 0, 1, 2. \quad (6.1)$$

W bardziej ogólnym przypadku funkcja wyjściowa może zależeć również od ostatniego przeczytanego znaku alfabetu A , tj. $w : S \times A \rightarrow B$.

Takie skończone automaty deterministyczne nazywamy *przekaźnikami*³ (ang. transducers):

Def. 6.2. *Skończonym automatem deterministycznym typu przekaźnika nazywamy następującą szóstkę: (A, B, S, s_0, p, w) , gdzie*

- A jest alfabetem wejściowym czytanym przez automat,
- B jest alfabetem wyjściowym pisany przez automat,
- S jest skończonym zbiorem stanów automatu,
- $s_0 \in S$ jest stanem początkowym,
- $p : A \times S \rightarrow S$ jest funkcją przejścia, przeprowadzającą automat ze stanu s w nowy stan s' , w zależności od przeczytanego znaku alfabetu.
- $w : S \times A \rightarrow B$ jest funkcją wyjściową, która podaje, jaki znak $b \in B$ ma wypisać automat, w zależności od przeczytanego symbolu oraz stanu, w jakim się znajduje.

Powyższy automat nazywamy *modelem Mealy'ego*⁴. W prostszym przypadku, gdy funkcja wyjścia nie zależy od czytanego znaku, $w : S \rightarrow B$, automat ten nazywamy *modelem Moora*⁵. Zauważmy, że przekaźniki nie definiują stanów akceptujących, gdyż są one teraz zastąpione ogólniejszym pojęciem funkcji wyjścia w .

Przekaźniki zazwyczaj wypisują wynik po przeczytaniu każdej nowej litery alfabetu wejściowego i przejściu do nowego stanu. Jeśli potraktujemy automat z rys. 6.1 jako model Moora z funkcją wyjścia (6.1), to liczba wejściowa $(1010)_2$ zostaje przetworzona na następujący ciąg: $(1, 2, 2, 1)$. Ostatnia liczba tego ciągu oznacza resztę z dzielenia liczby 10 przez 3, poprzednie liczby nie mają w przypadku sprawdzania podzielności znaczenia.

Więcej informacji o deterministycznych automatach skończonych (ang. *deterministic finite automaton* (DFA)) i podstawach *języków formalnych* można znaleźć np. w [53, 54].

³ Nazwa przekaźnik nie jest tu zbyt adekwatna i można pozostać przy terminach „model Mealy'ego” lub „model Moora”.

⁴ George H. Mealy (1927–2010), amerykański matematyk i informatyk.

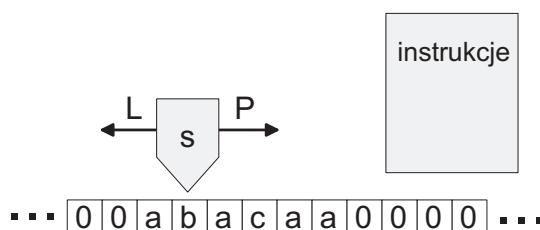
⁵ Edward F. Moore (1925–2003), amerykański matematyk i informatyk.

6.3 Maszyna Turinga

Algorytmy mogą być realizowane w bardzo różny sposób: jako programy komputerowe, poprzez układy elektroniczne (implementacja hardwarowa), z użyciem urządzeń mechanicznych (liczydło, arytmometr, czyli „kręciołek”), czy wreszcie „na piechotę”, poprzez wyliczanie ołówkiem na papierze. Do klasyfikacji algorytmów oraz badania teorii złożoności używamy jednak pewnego tworu abstrakcyjnego, mianowicie maszyny Turinga (TM). Na urządzenie to składają się następujące elementy (zob. rys. 6.2):

1. *Taśma*, podzielona na komórki ułożone jedna za drugą. Każda komórka zawiera symbol pewnego skończonego alfabetu A . Alfabet zawiera symbol pusty, 0. Początkowo taśma ma pewną liczbę komórek zapisaną symbolami ze zbioru $A \setminus \{0\}$. Niezapisane komórki taśmy zawierają symbol pusty. Ponadto taśma ma nieograniczoną długość w obie strony, co zapewnia jej nieograniczoną pamięć.
2. *Główica*, która może czytać i pisać symbole alfabetu A , oraz poruszać się w lewo (L) i w prawo (P) do sąsiednich komórek, zawsze o jeden krok w danym ruchu. Na początku głowica ustawiona jest na pierwszym od lewej niepustym symbolu.
3. *Rejestr stanów* – maszyna znajduje się zawsze w jednym stanie ze skończonego zbioru S . Stan początkowy to s_0 . Jeśli maszyna dojdzie do stanu ze zbioru $K \subset S$, zatrzymuje się.
4. *Tablica przejść*, definiująca co maszyna ma robić, gdy przeczyta dany symbol, będąc w określonym stanie. Tablica przejść określa, jaki symbol ma być zapisany, w którą stronę ma poruszyć się głowica (w lewo, L , lub w prawo, P) oraz w jaki nowy stan ze zbioru S ma przejść. Jeżeli tablica nie zawiera instrukcji dla danej kombinacji stanu i przeczytanego symbolu, maszyna zatrzymuje się.

Formalna definicja jest następująca:



Rysunek 6.2: Maszyna Turinga: nieskończona taśma podzielona jest na komórki zawierające symbole alfabetu, symbol 0 jest symbolem pustym. Głowica czytająca i pisząca symbole na taśmie jest w jednym z możliwych stanów s i może poruszać się w lewo lub w prawo. Instrukcje określają funkcję przejścia dla maszyny deterministycznej albo relację przejścia dla maszyny niedeterministycznej

Tabela 6.1: Funkcja przejścia przykładowej maszyny Turinga

Stan	Symbol przeczytany	Symbol zapisany	Ruch głowicy	Nowy stan
s_0	1	\rightarrow	0	P
s_1	0	\rightarrow	0	P
s_1	1	\rightarrow	1	P
s_2	0	\rightarrow	1	L
s_2	1	\rightarrow	1	P
s_3	0	\rightarrow	0	L
s_3	1	\rightarrow	1	L
s_4	0	\rightarrow	1	P
s_4	1	\rightarrow	1	L

Def. 6.3. *Deterministyczną maszyną Turinga (DTM) o jednej taśmie nazywamy szóstkę $(A, 0, S, s_0, K, p)$, gdzie*

- A jest skończonym alfabetem,
- $0 \in A$ jest symbolem pustym,
- S jest skończonym zbiorem stanów,
- $s_0 \in S$ jest stanem początkowym,
- $K \subset S$ jest zbiorem stanów akceptujących,
- $p : A \times S \rightarrow A \times S \times \{L, P\}$ jest funkcją przejścia.

Termin *deterministyczna* oznacza, że każdy kolejny krok maszyny jest ściśle określony na podstawie jej aktualnego stanu i przeczytanego symbolu.

Jako bardzo prosty przykład rozważmy DTM, która przeprowadza taśmę ze stanu typu

...000011110000....

w stan

...000111101111000...,

tj. kopiuje ciąg n jedynek po prawej stronie od początkowego ciągu, oddzielając stary i nowy ciąg symbolem 0. Symbole 0 rozciągające się w lewo i w prawo do nieskończoności to zgodnie z definicją niezapisane fragmenty taśmy. Tak więc $A = \{0, 1\}$. Maszynka ma 5 stanów, $S = \{s_0, s_1, s_2, s_3, s_4\}$. Zdefiniujmy funkcje przejścia p zgodnie z tabelą 6.1. Prześledźmy kolejne kroki pracy zdefiniowanej powyżej maszyny. Położenie głowicy będziemy oznaczać wytłuszczeniem symbolu na taśmie. Zaczynamy, zgodnie z definicją, z głowicą maszyny ustawionej na pierwszym z lewej symbolu 1. Tabela 6.2 przedstawia sekwencję kroków, z aktualnym zapisem taśmy oraz stanem maszyny. Po 14 krokach maszynka zatrzymuje się, ponieważ w tabeli 6.1 nie ma instrukcji dla stanu s_0 i przeczytanego symbolu 0 – ta sekwencja oznacza STOP. Widzimy, że w wyniku działania istotnie został

Tabela 6.2: Kolejne kroki przykładowej maszyny Turinga

Krok	Taśma	Stan
0	...000011000000...	s_0
1	...000001000000...	s_1
2	...000001000000...	s_1
3	...000001000000...	s_2
4	...000001010000...	s_3
5	...000001010000...	s_4
6	...000001010000...	s_4
7	...000011010000...	s_0
8	...000010010000...	s_1
9	...000010010000...	s_2
10	...000010010000...	s_2
11	...000010011000...	s_3
12	...000010011000...	s_3
13	...000010011000...	s_4
14	...000011011000...	s_0

skopiowany ciąg jedynek. Można się przekonać, że wykonanie algorytmu z inną liczbą jedynek też da pożądany rezultat.

Istnieje bardzo wiele wariantów maszyn Turinga: wielotaśmowa, wielogłówkowa, dopuszczająca pozostanie głowicy w tej samej pozycji, z taśmą ograniczoną z lewej strony, czytająca i pisząca po kilka symboli w jednym kroku, posiadająca wiele rejestrów itd. Okazuje się jednak, że wszystkie te warianty są równoważne zwykłej maszynie Turinga.

Śledząc działanie naszej przykładowej TM, nie można oprzeć się wrażeniu, że mechanizm jest bardzo skomplikowany i cały algorytm nie jest praktyczny. Jest tak istotnie. Nie buduje się użytkowych maszyn Turinga – mają one znaczenie czysto teoretyczne, ale fundamentalne. Niezwykle istotnym pojęciem jest tzw. *uniwersalna maszyna Turinga* (UTM). Jest to maszyna, która może modelować każdą inną maszynę Turinga. Aby to zrozumieć, uświadommy sobie, że modelowanie komputera przez komputer nie jest niczym niezwykłym: możemy odpalić aplikację kalkulatora na PC-cie, czyli „duży” komputer emuluje mały kalkulatork. W przypadku maszyn Turinga UTM odczytuje najpierw z taśmy zestaw instrukcji (jak np. ten z tabeli 6.1) specyficzny dla danej emulowanej maszyny, a następnie wykonuje algorytm, de facto realizując rachunek tej specyficznej maszyny. Co więcej, UTM wcale nie musi być duża w sensie liczbowości zbiorów A i S : najmniejsza znana UTM ma zaledwie 2 stany i 5 symboli [55].

Istota teoretycznego znaczenia maszyn Turinga tkwi w tzw. *hipotezie Turinga-Churcha*⁶ mówiącej, że każdy możliwy rachunek, każdy algorytm może

⁶ Alonzo Church (1903-1995), amerykański matematyk i logik, jeden z twórców podstaw informatyki, jego studentem był m.in. Turing.

być wykonany przez UTM. Wszystkie uniwersalne maszyny Turinga są sobie równoważne i stanowią najbardziej ogólny *model obliczeniowy*. Po prostu wszystko, co możemy w ogóle obliczyć, możemy obliczyć na UTM. Komputery z użytkowymi językami programowania są właśnie konkretnymi realizacjami uniwersalnych maszyn Turinga.

Bardzo głębokim zastosowaniem TM jest dowód twierdzenia Gödla⁷ o nierozstrzygalności prawdziwości zdąń w nietrywialnych teoriach matematycznych (np. w systemach zawierających arytmetykę liczb naturalnych). W kontekście maszyn Turinga zagadnienie to przejawia się jako *problem zatrzymywania się* (ang. *halting problem*). Wyobraźmy sobie TM czytającą jakieś zdanie i mającą stwierdzić, czy jest prawdziwe, czy nie. Jeśli maszyna skończy pracę w stanie odpowiadającemu prawdziwe lub fałszowi, otrzymujemy rozstrzygnięcie. Maszyna może jednak działać w nieskończoność i nigdy nie zakończyć pracy. Turing pokazał, że *nie ma uniwersalnego algorytmu pozwalającego stwierdzić, czy dwolna maszyna Turinga zakończy pracę*, tym samym w obrębie danego systemu formalnego (języka) istnieją zdania nierozstrzygalne. Zainteresowani tą bardziej filozoficzno-matematyczną tematyką odsyłani są do bardzo przystępnego, a precyzyjnego wyjaśnienia w książce „Nowy umysł cesarza” Penrose'a⁸ [56].

Przy klasyfikacji algorytmów używamy również pojęcia niedeterministycznej maszyny Turinga (NDTM), będącej uogólnieniem o następującej formalnej definicji:

Def. 6.4. *Niedeterministyczną maszyną Turinga (NDTM) o jednej taśmie nazywamy szóstkę $(A, 0, S, s_0, K, R)$, gdzie*

- A jest skończonym alfabetem,
- $0 \in A$ jest symbolem pustym,
- S jest skończonym zbiorem stanów,
- $s_0 \in S$ jest stanem początkowym,
- $K \subset S$ jest zbiorem stanów akceptujących,
- R jest relacją $(A \times S) \times (A \times S \times \{L, P\})$ określającą możliwe przejścia ze zbioru $(A \times S)$ w zbiór $(A \times S \times \{L, P\})$.

Różnica w stosunku do maszyny deterministycznej tkwi w określeniu przejść, zapisywanego symbolu i kierunku ruchu głowicy, które w NDTM nie jest jednoznaczne, dlatego w ostatnim punkcie definicji mamy *relację*, a nie funkcję. Maszyna niedeterministyczna ma więc do wyboru w każdym kroku kilka możliwości „dalszego rozwoju”. Przykładowa tabela dla NDTM może wyglądać następująco:

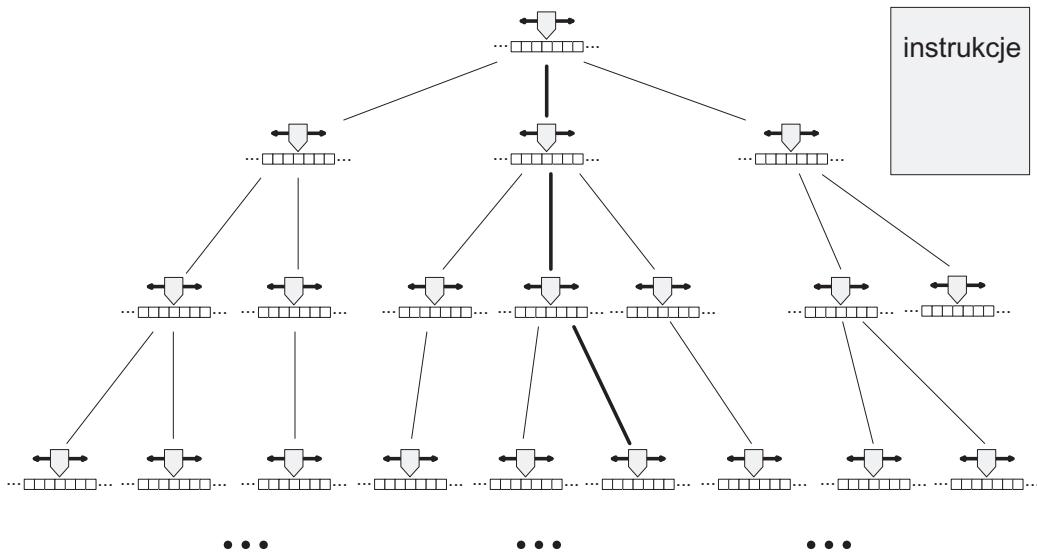
⁷ Kurt Gödel (1906-1978), austriacki logik, matematyk i filozof, jego słynne twierdzenie o nierozstrzygalności zmieniło rozumienie podstaw matematyki.

⁸ Sir Roger Penrose (1931), brytyjski fizyk, kosmolog i matematyk.

Stan	Symbol przeczytany	Symbol zapisany	Ruch głowicy	Nowy stan
		...		
		1	P	s_1
s_2	1	→	0	L
			0	s_2
			P	s_4
		...		

Para $(s_2, 1)$ jest w relacji z trzema trójkami: $(1, P, s_1)$, $(0, L, s_2)$ i $(0, P, s_4)$, więc przejście może nastąpić do każdej z tych możliwości.

Działanie takiej maszyny możemy interpretować na dwa sposoby: *probabilistyczny* i *multiplikatywny* (lub „rozmnożeniowy”). W pierwszej interpretacji NDTM losuje w każdym kroku, którą z możliwości będących w relacji z parą (przeczytany symbol, stan) wybrać, tzn. do jakiego nowego stanu przejść, jaki symbol zapisać na taśmie oraz w którą stronę przesunąć głowicę. Maszyna jest więc cały czas jedna i ta sama, natomiast może wybrać wiele „ścieżek” podczas wykonywania obliczeń. W drugiej interpretacji, *multiplikatywnej*, maszyna rozdziela się w pierwszym kroku na kilka NDTM (tyle, ile jest potrzebnych, by wysyścić wszystkie możliwości dopuszczane przez relację przejścia R), w kolejnym kroku każda z tych nowych maszyn rozdziela się na następne itd., patrz rys. 6.3. Mamy więc de facto do czynienia z bardzo wieloma maszynami, które pracują



Rysunek 6.3: Multiplikatywna interpretacja niedeterministycznej maszyny Turinga. W każdym kroku maszyna rozmnaża się na tyle kopii, ile jest potrzebnych, by wysyścić wszystkie możliwości dopuszczane przez relację przejścia R podaną w instrukcji. W kolejnym kroku każda z tych nowych maszyn rozdziela się na następne itd. Maszyny pracują równolegle. W interpretacji probabilistycznej maszyna ewoluje jedną ścieżką obliczeń, przykładowo zaznaczoną pogrubioną kreską. W każdym kroku odbywa się losowanie, którą możliwość wybrać

równolegle, a z kroku na krok jest ich coraz więcej. Czas obliczeń jest proporcjonalny do liczby poziomów na rys. 6.3, ale koszt obliczeń jest proporcjonalny do sumarycznego czasu pracy wszystkich maszyn Turinga. Właśnie tę multiplikatywną interpretację wygodnie jest mieć na uwadze przy omawianiu klasyfikacji problemów decyzyjnych w rozdz. 6.6.

W interpretacji probabilistycznej wybieramy tylko jedną możliwą ścieżkę obliczeń, np. tę zaznaczoną pogrubioną kreską na 6.3. Maszyna jest więc jedna, a w każdym kroku odbywa się losowanie, którą ścieżkę wybrać do dalszego ciągu obliczeń.

6.4 Problem 196

Powróćmy teraz do *problemu zatrzymywania*. Przykładem bardzo prostego algorytmu, o którym nie wiadomo, czy się kończy, czy nie, jest tzw. *problem „196”*, należący do zagadnień matematyki rekreacyjnej. Zdefiniujmy najpierw *liczby przenicowane*:

Def. 6.5. *Liczba przenicowaną do liczby naturalnej n jest liczba otrzymana przez odwrócenie kolejności cyfr n .*

Mówiąc inaczej, czytamy liczbę wspak, np. liczbą przenicowaną do 125 jest 521.

Def. 6.6. *Liczba palindromiczną jest liczba, która jest równa swojej liczbie przenicowanej (czyta się tak samo wprzód i wspak).*

Na przykład liczba 64146 jest palindromiczna. Mając te definicje, utwórzmy następujący algorytm:

Alg. 6.1 (algorytm „196”).

1. Do liczby naturalnej n dodaj jej liczbę przenicowaną i oznacz sumę jako n .
2. Jeśli n jest palindromiczną, STOP. Jeśli nie, idź do kroku 1.

Po prostu bierzemy daną liczbę naturalną i dodajemy liczbę do niej przenicowaną tak długo, aż dostaniemy w wyniku liczbę palindromiczną. Dla małych liczb procedura ta kończy się bardzo szybko, np. $17 + 71 = 88$ itp. Więcej pracy wymaga np. liczba $n = 96$:

$$96 + 69 = 165, \quad 165 + 561 = 726, \quad 726 + 627 = 1353, \quad 1353 + 3531 = 4884.$$

W miarę zwiększania wyjściowej liczby n wszystko idzie cacy-cacy, aż dochodzimy do liczby 196. Zaczynając Algorytm 196 od tej liczby dostajemy w kolejnych iteracjach ciąg 196, 887, 1675, 7436, 13783, 52514, ..., który wydaje się nie kończyć, tzn. nie daje liczby palindromicznej. Do chwili obecnej sprawdzono komputerowo, że nawet po 600 milionach iteracji algorytm się nie kończy. Nie ma

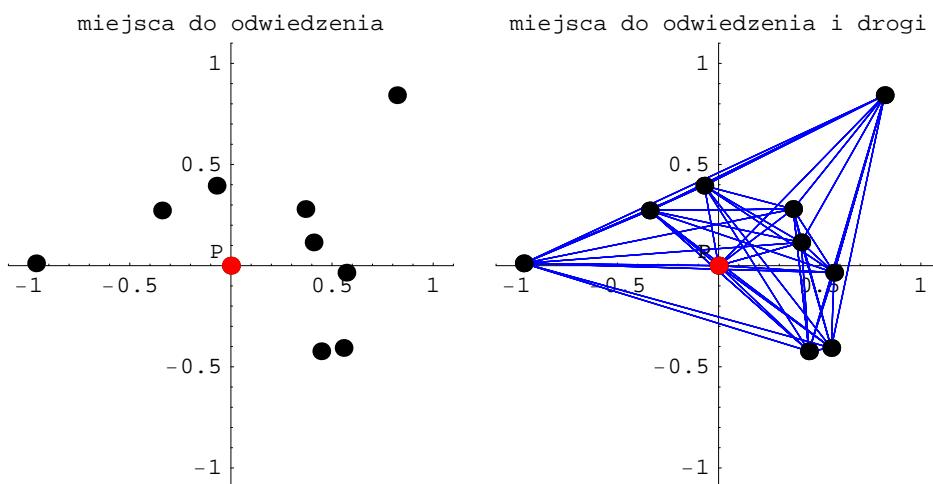
jednak dowodu, że się nie kończy nigdy! Tak więc mamy bardzo prosty algorytm, o którym (na dziś) nie wiemy, czy się kończy, czy nie. W ten sposób rekrecyjny problem „196” awansował do grona najsłynniejszych nieroziążanych problemów matematycznych.

Następne liczby, o których nie wiadomo, czy dadzą w wyniku zastosowania algorytmu „196” liczbę palindromiczną, to 295, 394, 493, 592, 689, 691 ...

6.5 Lodziarz z Pińczowa i algorytm genetyczny

Przedstawimy teraz słynny problem, który jest problemem trudnym, czyli czas obliczeń rośnie bardzo szybko z długością danych (jest klasy NPC). Jest to tzw. *problem komiwojażera*. Aby dodać lokalnego kolorytu, nazwijmy go lodziarzem z Pińczowa. Otóż sprzedając towar, lodziarz ma odwiedzić okoliczne miasteczka i wsie w taki sposób, aby wszystkie obejść oraz aby sumaryczna pokonana droga była jak najmniejsza. Jest to więc problem optymalizacyjny, czyli szukający najlepszego rozwiązania. Zakładamy też dla prostoty, że lodziarz porusza się po liniach prostych między miejscowościami oraz że wraca wieczorem do domu!

Załóżmy, że miasteczek do odwiedzenia jest n i oczywiście znamy ich wzajemne odległości. Sytuację przedstawia rys. 6.4, gdzie Pińczów jest w początku układu współrzędnych, a pozostałe punkty oznaczają miejscowości, które należy odwiedzić. W naszym przykładzie współrzędne tych punktów zostały wygenerowane losowo, każda w przedziale $[-1, 1]$. Na rysunku po prawej stronie dodaliśmy połączenia między miejscowościami. Odległości są w naszym przykładzie po prostu euklidesowymi odległościami między punktami na płaszczyźnie, czyli dystans między i -tym i j -tym miasteczkiem wynosi $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, gdzie x i y są współrzędnymi na płaszczyźnie. W ogólnym problemie komiwojażera



Rysunek 6.4: Lewa strona: rozmieszczenie miejscowości, które musi odwiedzić lodziarz z Pińczowa. Prawa strona: to samo z dorysowanymi drogami łączącymi miejscowości

odległości te (wagi) mogą być dowolnymi liczbami dodatnimi. Problem naszego lodziarza na pozór nie wygląda jakoś szczególnie skomplikowanie. Okazuje się jednak, że nie znamy efektywnego, czyli szybkiego, algorytmu rozwiązania tego problemu.

Zgodnie z poznaną nomenklaturą teorii grafów mamy odpowiedniość

miejscowość	wierzchołek grafu
linia łącząca miejscowości	krawędź grafu
odległość między miejscowościami	waga krawędzi grafu
droga zamknięta przechodząca przez	
każdą miejscowością dokładnie raz	cykl Hamiltona
rozwiązywanie problemu komiwojażera	cykl Hamiltona o najmniejszej wadze

Musimy więc znaleźć *cykl Hamiltona* dla grafu z prawej strony rys. 6.4 o najmniejszej sumarycznej wadze.

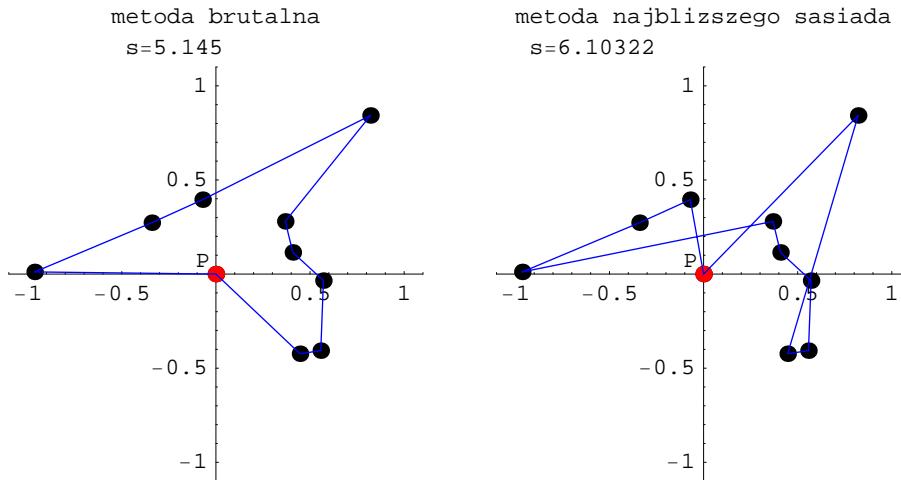
Dlaczego ten problem jest trudny? Zastanówmy się najpierw, ile jest wszystkich możliwych dróg. Pierwsze miasteczko lodziarz może wybrać na n sposobów, następnie na $n - 1$, kolejne na $n - 2$, a więc wszystkich możliwości jest $n(n - 1)(n - 2) \dots 3 \cdot 2 \cdot 1 = n!$ (silnia). Możemy jeszcze utożsamić drogi obchodzone w odwrotnych kierunkach, wtedy mamy $n!/2$ możliwości, np. dla $n = 20$ daje to $20!/2 = 1216451004088320000$ dróg! Jest to liczba astronomiczna nawet dla stosunkowo małego n , co powoduje, że użycie tzw. metody *brutalnej siły* nie jest możliwe dla dużych n . Metoda brutalnej siły polega bowiem na sprawdzeniu po kolei wszystkich możliwości i wybraniu najlepszej. Podkreślimy, że metoda ta zawsze działa, ale jeśli n jest dostatecznie duże to na jej przeprowadzenie po prostu może nie stać nam czasu czy pieniędzy na zakup superkomputerów (pewną oszczędność daje tu zastosowanie algorytmu z powracaniem).

Na rys. 6.5 przedstawiony jest wynik metody brutalnej siły dla przypadku $n = 9$. Wszystkie $9!/2 = 181440$ możliwości sprawdził komputer (zwykły peçet tu spokojnie wystarcza), dla każdej policzył sumaryczną drogę, po czym wybrał najlepszą możliwą, czyli tę ukazaną na rysunku. Długość drogi (waga) wynosi w tym przypadku $s = 5.145$ i jest to wynik nie do pobicia!

A teraz bardzo szybki algorytm typu *zachłannego*, def. 6.7.

Def. 6.7. *Algorytm zachłanny to algorytm, który w każdym kroku kieruje się „doraźną korzyścią”, nie wybiegając wprzód i nie sprawdzając, czy podejmowana decyzja jest globalnie najlepsza.*

A więc algorytm zachłanny w każdym kroku podejmuje decyzję w danym momencie najlepszą, nie martwiąc się o przyszłość i „globalną optymalizację”. W tym przypadku jest to algorytm *najbliższego sąsiada*. W każdej miejscowości lodziarz sprawdza, które z nieodwiedzonych jeszcze miejsc jest aktualnie najbliższe i tam się udaje, stąd nazwa. Wynik pokazany jest po prawej stronie rys. 6.5. W tym przypadku $s = 6.1$, więc o ok. 20% gorzej niż dla metody brutalnej



Rysunek 6.5: Lewa strona: metoda brutalnej siły. Prawa strona: metoda najbliższego sąsiada

siły. Porównując dwie drogi, widzimy, gdzie lodziarz stracił. Najpierw „niepotrzebnie” poszedł do góry zamiast w lewo, ponadto ostatni odcinek drogi jest bardzo długi. Jest to zresztą typowa przypadłość tego algorytmu: zasada najbliższego sąsiada często ustala trasę w taki sposób, że po przedostatnim kroku wędrowiec jest daleko od domu! Natomiast algorytm jest błyskawiczny: na starcie lodziarz sprawdza n miejscowości, potem $n - 1$ itd., łącznie algorytm wykonuje więc $\sum_{i=1}^n i = n(n + 1)/2$ operacji, dla $n = 9$ jest to zaledwie 45 w porównaniu z astronomiczną liczbą operacji algorytmu brutalnej siły.

Nie jest znany algorytm rozwiązujący problem komiwojażera, czyli znajdujący *minimalny* cykl Hamiltona danego grafu, w czasie wielomianowym, tzn. rosnącym z n nie szybciej niż n^m , gdzie m jest pewną liczbą naturalną niezależną od n . Algorytm najbliższego sąsiada działa co prawda w czasie wielomianowym $\sim n^2$, ale nie znajduje rozwiązania problemu, jedynie pewne przybliżenie, nawet nie wiadomo jak dobre.

Czy istnieje algorytm pośredni, czyli rozsądnie szybki i jednocześnie w miarę dokładny? Odpowiedź jest twierdząca, a takie algorytmy nazywamy *heurystycznymi*. Nie znajdują one najlepszego rozwiązania, ale dają zupełnie niezłe, co więcej, w krótkim czasie. Przestawimy tu po pokrótce ideę i działanie tzw. *algorytmów genetycznych* [57], będących przykładem algorytmów heurystycznych. Każda trasa lodziarza może być przedstawiona w postaci ciągu liczb naturalnych, gdzie każda liczba to numer odwiedzonej miejscowości. Jest tu analogia do sekwencji genów. Rozważmy ponownie przykład z rys. 6.4. Możliwe trasy-genotypy to np.

$$t_1 = (5 \ 8 \ 1 \ 9 \ 7 \ 2 \ 6 \ 3 \ 4),$$

$$t_2 = (8 \ 5 \ 9 \ 1 \ 4 \ 3 \ 7 \ 2 \ 6),$$

...

z długościami dróg, odpowiednio, $s_1 = 6.38$, $s_2 = 6.25$. Pierwszym etapem algorytmu genetycznego jest wygenerowanie pewnej liczby N tras – jest to pokolenie praprzodków. Sposób generowania jest nieistotny, możemy po prostu wykreować przypadkowo ciągi liczb. Następnie praprzodkowie pobierają się i rozmnażają, poczynając dzieci, które część informacji genetycznej dziedziczą po ojcu, a część po matce. W szczególności t_1 i t_2 płodzą dzidziusia, u którego np. pierwsze cztery geny pochodzą od t_1 , a następne pięć od t_2 , czyli mamy podział

$$\begin{aligned} t_1 &= (5 \ 8 \ 1 \ 9 \mid 7 \ 2 \ 6 \ 3 \ 4), \\ t_2 &= (8 \ 5 \ 9 \ 1 \mid 4 \ 3 \ 7 \ 2 \ 6), \end{aligned}$$

dający w efekcie

$$d_1 = (5 \ 8 \ 1 \ 9 \mid 4 \ 3 \ 7 \ 2 \ 6), \quad s = 5.54.$$

Wzięcie sześciu genów od t_1 i trzech od t_2 dałoby $(5 \ 8 \ 1 \ 9 \ 7 \ 2 \mid 7 \ 2 \ 6)$. Jest to kombinacja nielegalna, bo 2 i 7 występują podwójnie, a brakuje 3 i 4 (miejscowości 3 i 4 byłyby nieodwiedzone). W takiej sytuacji naprawiamy genotyp, zastępując powtarzające się allele⁹ brakującymi, dostając np.

$$d_2 = (5 \ 8 \ 1 \ 9 \ 7 \ 2 \mid 4 \ 3 \ 6), \quad s = 6.79.$$

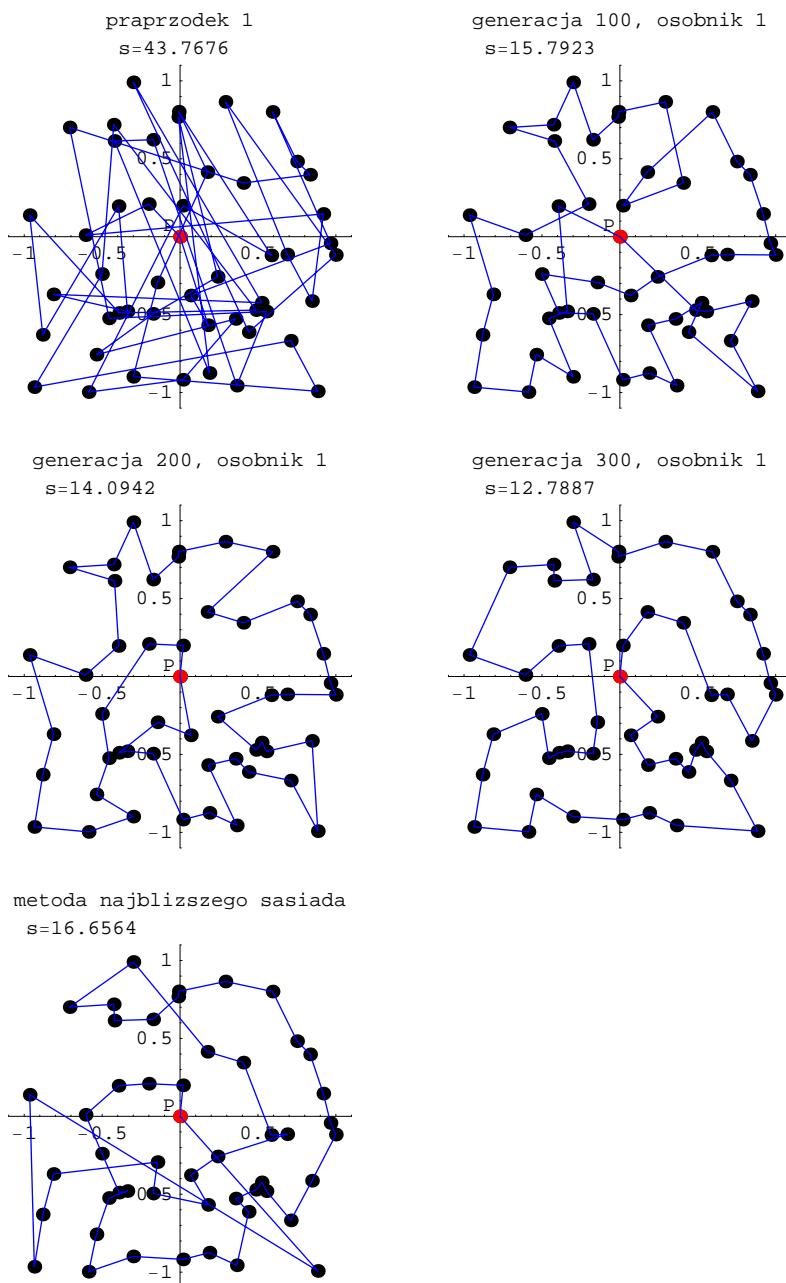
Czasami korzystne jest odwrócenie kolejności genów w jednej części dziedziczonego genotypu, co w naszym przykładzie daje

$$d_3 = (5 \ 8 \ 1 \ 9 \mid 6 \ 2 \ 7 \ 3 \ 4), \quad s = 6.30.$$

Oprócz wyżej przedstawionego krzyżowania genotypów możliwe są jeszcze *mutacje*, czyli losowe niewielkie zmiany genotypu u dziecka. Polegają one na przedstawieniu dwóch genów. Zmieniając miejscami czwarty i piąty gen (wytluszczone) w dziecku d_1 , dostajemy $d_4 = (5 \ 8 \ 1 \ 9 \ 2 \ 7 \ 4 \ 3 \ 6)$ itd. Po wygenerowaniu dużej liczby dzieci, rzędu N^2 (możemy w tym celu krzyżować każdego z każdym!), stosujemy darwinowską zasadę doboru naturalnego. Przeżywają tylko te dzieci, które dają N najmniejszych wartości s . Następnie dzieci stają się rodzicami dla kolejnego pokolenia itd. Wielkość populacji utrzymujemy na tym samym poziomie N .

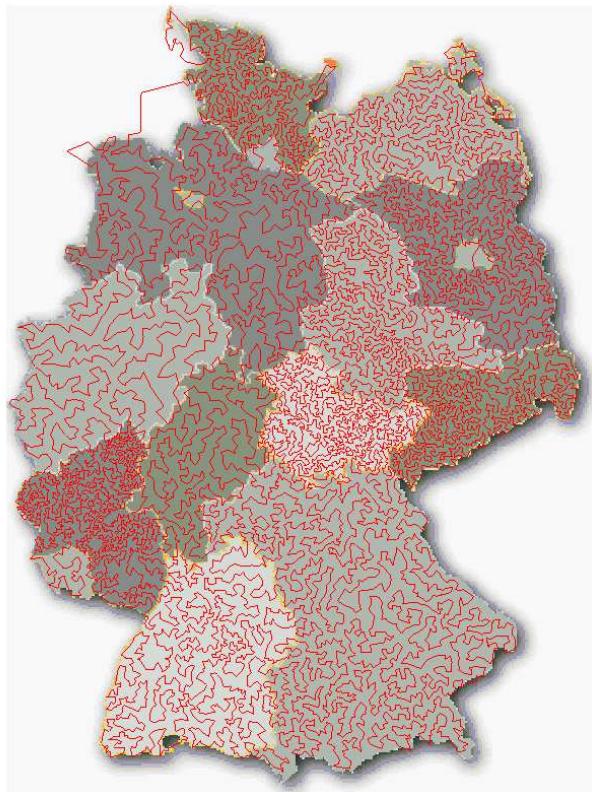
Rysunek 6.6 ilustruje działanie algorytmu genetycznego dla przykładu z $n = 49$. W górnym lewym rogu przedstawiony jest typowy praprzodek, jeszcze niedostosowany do pracy lodziarza, z bardzo zyzkakową drogą. Po 100 generacjach (górnny prawy róg) droga wygląda już znacznie bardziej regularnie, co więcej, jest bardziej optymalna niż w algorytmie najbliższego sąsiada, co widzimy z porównania wartości s . Dalsza ewolucja skracą wartości s , w efekcie w pokoleniu 400 dostajemy ładnie wyglądającą drogę, bez zyzkaków, przecież, które zginęły w mrokach dziejów. Czy jest to najlepsza droga? Tego niestety

⁹ Allel to wartość genu.



Rysunek 6.6: Algorytm genetyczny w zastosowaniu do problemu komiwojażera. Kolejne cztery wykresy obrazują ewolucję od praprzodka do generacji nr 300 (pokazujemy tylko jednego osobnika z każdego pokolenia). Już po mniej niż 100 generacjach droga s staje się krótsza niż w algorytmie najbliższego sąsiada (dolny rysunek)

nie wiemy, ale jest na pewno lepsza od drogi z algorytmu najbliższego sąsiada. Ponadto, algorytm jest szybki, oferując kompromis między jakością i szybkością. W pokazanym rachunku wykorzystaliśmy $N = 30$ i pozwoliliśmy każdej parze na czwórkę dzieci, co dało łączną liczbę kroków do pokolenia 400



Rysunek 6.7: Dokładne rozwiązywanie problemu niemieckiego komiwojażera dla 15112 miast. Optymalna trasa ma długość ok. 66000 km
 $(\text{http://www.math.uwaterloo.ca/tsp/history/pictorial/d15112.html})$

równą jedynie $400 \cdot N^2 = 1440000$, czyli zgoła nic w porównaniu z $49!/2 = 304140932017133780436126081660647688443776415689605120000000000$.

Dlaczego metoda działa dla problemu lodziarza? Nie jest to w pełni zrozumiałe od strony teoretycznej. Potoczne wyjaśnienie mówi, że jeśli ojciec jest trasą optymalną w jednej części, a matka w innej, to trasa-dziecko może odziedziczyć najlepsze geny po ojcu i po matce (jaki śliczny!) i jest wówczas lepsza w obu częściach.

W ostatnich latach algorytmy genetyczne są bardzo intensywnie rozwijane i znajdują zastosowanie w wielu procesach optymalizacyjnych. Ich zachowanie jest fascynujące w swoim imitowaniu procesu ewolucji, opartego na dziedziczeniu, krzyżowaniu genotypów, mutacjach, wreszcie stosowaniu zasady doboru naturalnego. Algorytmy te zostały spopularyzowane przez przez Hollandę¹⁰ w 1970 r. [58].

Kończymy ten rozdział przedstawieniem (rys. 6.7) dokładnego rozwiązywania problemu komiwojażera dla 15,112 niemieckich miejscowości. Rachunek wykonano w 2001 r. na uniwersytetach w Rice i Princeton przy użyciu pracujących

¹⁰ John Henry Holland (1929-), amerykański psycholog i informatyk.

równolegle przez 11 tygodni 110 procesorów o taktowaniu 500 MHz. Zwróćmy uwagę, że trasa nie przecina się (zob. zad. 6.5).

Najszybszy znany algorytm dokładnego rozwiązywania problemu komiwojażera o n miastach, oparty na tzw. programowaniu dynamicznym [4, 59], ma złożoność $\mathcal{O}(n^2 2^n)$.

Obecnie rekordowej wielkości problem komiwojażera, dokładnie rozwiązyany w latach 2005/2006, zawiera 85900 „miast”. Są one bramkami logicznymi na płytce układu scalonego, łączonymi ścieżkami wypalanymi przez laser www.math.uwaterloo.ca/tsp/pla85900/index.html.

6.6 Klasyfikacja złożoności problemów decyzyjnych

Zaczniemy od zdefiniowania co abstrakcyjnie rozumiemy przez *problem*.

Def. 6.8. Problem to odpowiednio określona relacja zbioru \mathcal{P} przypadków problemu oraz zbioru \mathcal{R} rozwiązań problemu.

Dla problemu komiwojażera zbiór \mathcal{P} to zbiór wszystkich grafów z nieujemnymi wagami, a zbiór \mathcal{R} to zbiór ciągów wierzchołków. Określenie relacji to „dla danego grafu znajdź cykl Hamiltona o najmniejszej wadze”. Dany przypadek problemu (konkretny graf) jest w relacji z ciągiem wierzchołków określającym optymalną drogę komiwojażera. Ciąg ten może być pusty, jeśli graf nie ma cyklu Hamiltona (np. graf niespójny) – wtedy dany przypadek problemu nie ma rozwiązania. Dany graf może też być w relacji z więcej niż jednym ciągiem, jeśli drogi odpowiadające tym ciągom mają tę samą długość – wtedy dla danego przypadku istnieje kilka rozwiązań.

Def. 6.9. Problem rozstrzygalny to problem, dla którego istnieje algorytm, który dla każdego przypadku problemu znajduje rozwiązanie, bądź stwierdza, że dany przypadek rozwiązania nie ma. Problem jest nierozstrzygalny, jeśli istnieją takie przypadki w zbiorze \mathcal{R} , dla których nie da się stwierdzić, czy są w relacji z jakimś elementem zbioru \mathcal{P} , czy też nie.

Za wyjątkiem problemu „196”, dyskutowane w tym wykładzie problemy są rozstrzygalne. Zauważmy, że dla problemu nierozstrzygalnego możemy być w stanie znaleźć rozwiązania dla pewnych przypadków.

Algorytm skonstruowany do rozwiązywania problemu rozstrzygalnego musi rozwiązywać wszystkie jego przypadki. Podstawową charakterystyką algorytmu jest czas jego wykonywania dla danego przypadku. W oczywisty sposób czas ten rośnie z długością danych wejściowych (czy liczb użytych do opisu przypadku), ale wzrost ten może być bardzo różny. Na przykład czas wykonywania algorytmu sprawdzającego podzielność liczby binarnej przez 3 z rozdz. 6.2 jest proporcjonalny do liczby cyfr wejściowej liczby, n , czyli rośnie wolno z n , podczas gdy

brutalna metoda rozwiązywania problemu komiwojażera z rozdz. 6.5 rośnie jak silnia liczby miejscowości, $n!$, czyli bardzo szybko.

Tak więc wzrost długości danych problemu bezpośrednio przekłada się na wzrost czasu rachunku. Aby nasze oszacowania miały bardziej namacalny charakter, założymy, że nasz komputer wykonuje 10^9 operacji na sekundę, czyli wykonanie algorytmu z danymi o długości $n = 1$ trwa $t_0 = 10^{-9}$ s. Przykład czasu obliczeń dla poszczególnych wartości n oraz różnych typów algorytmów przedstawiony jest w tabeli 6.3. Pierwsza kolumna zawiera długość danych n , zwiększącą się w dół tabeli. Druga kolumna zawiera czas t wykonywania algorytmu, który rośnie liniowo z długością danych n , czyli $t = t_0 n$. W tym przypadku czas wykonywania, nawet dla dużych n , są bardzo krótkie. W trzeciej kolumnie zakładamy, że $t = t_0 n \log n$. Widzimy, że przemnożenie przez „wolno zmienny” logarytm ma bardzo mały efekt. W czwartej kolumnie mamy tzw. wzrost wielomianowy, $t = t_0 n^p$, z niskim wykładnikiem $p = 2$. W tym przypadku również wzrost jest dość powolny i bez trudu możemy przeprowadzić algorytm nawet dla dużych wartości n . Sytuacja zmienia się dla wzrostu wielomianowego z dużym wykładnikiem, $p = 10$ (piąta kolumna), gdzie dla $n = 100$ musielibyśmy czekać na wynik 3000 lat, dla $n = 200$ aż 3 mln lat itd. Wzrost wykładniczy, $t = t_0 2^n$ (szósta kolumna), jest jeszcze szybszy, wzrost jak $n!$ jeszcze silniejszy, wzrost jak n^n silniejszy od silni. Oczywiście możemy sobie wyobrazić jeszcze silniejsze wzrosty, np. jak n^{n^n} . Rzecz leży w tym, że wzrosty silniejsze od potęgowego są tak szybkie, że już dla danych długości $n \simeq 100$ czas wykonywania obliczeń jest dłuższy od wieku Wszechświata, który według obecnych oszacowań wynosi 13.7 mld lat! Takie algorytmy są więc zupełnie bezużyteczne dla dłuższych danych.

Wprowadza się następujący przybliżony podział:

Tabela 6.3: Czas wykonywania algorytmów o różnej złożoności (pierwszy wiersz tabeli) w zależności od długości danych, n , przy założeniu, że komputer wykonuje 10^9 elementarnych operacji na sekundę. Symbol s oznacza sekundy, g – godziny, d – dni, l – lata. Wiek Wszechświata to $13.7 \cdot 10^9$ lat

n	n	$n \log n$	n^2	n^{10}	2^n	$n!$	n^n
2	$0.002 \mu\text{s}$	$0.001 \mu\text{s}$	$0.004 \mu\text{s}$	$1 \mu\text{s}$	$0.004 \mu\text{s}$	$0.002 \mu\text{s}$	$0.004 \mu\text{s}$
5	$0.005 \mu\text{s}$	$0.008 \mu\text{s}$	$0.03 \mu\text{s}$	10 ms	$0.03 \mu\text{s}$	$0.1 \mu\text{s}$	$3 \mu\text{s}$
10	$0.01 \mu\text{s}$	$0.02 \mu\text{s}$	$0.1 \mu\text{s}$	10 s	$1 \mu\text{s}$	4 ms	10 s
20	$0.02 \mu\text{s}$	$0.06 \mu\text{s}$	$0.4 \mu\text{s}$	3 g	1 ms	77 l	$3 \cdot 10^{10} \text{ l}$
50	$0.05 \mu\text{s}$	$0.2 \mu\text{s}$	$3 \mu\text{s}$	3 l	13 d	10^{47} l	$3 \cdot 10^{68} \text{ l}$
100	$0.1 \mu\text{s}$	$0.5 \mu\text{s}$	$10 \mu\text{s}$	3 tys. l	$4 \cdot 10^{13} \text{ l}$	$3 \cdot 10^{141} \text{ l}$	$3 \cdot 10^{183} \text{ l}$
200	$0.2 \mu\text{s}$	$1 \mu\text{s}$	0.04 ms	$3 \cdot 10^6 \text{ l}$	$5 \cdot 10^{43} \text{ l}$	$3 \cdot 10^{358} \text{ l}$	$5 \cdot 10^{443} \text{ l}$
500	$0.5 \mu\text{s}$	$3 \mu\text{s}$	0.3 ms	$3 \cdot 10^{10} \text{ l}$	10^{134} l	$3 \cdot 10^{1117} \text{ l}$	10^{1333} l

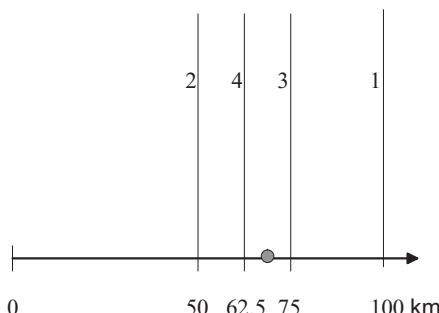
Def. 6.10. *Algorytmy szybkie to algorytmy, dla których czas wykonywania rośnie wielomianowo z długością danych, przy czym stopień wielomianu nie jest zbyt wysoki. Algorytmy wolne to algorytmy rosnące wykładniczo lub szybciej.*

Problemy, z którymi się spotykamy, można podzielić na decyzyjne oraz optymalizacyjne. W problemach decyzyjnych odpowiadą jest *tak* lub *nie*.

Def. 6.11. *Problem decyzyjny to taki problem, w którym zbiór $\mathcal{R} = \{\text{tak}, \text{nie}\}$. Inaczej, odpowiedź na każdy przypadek problemu brzmi tak lub nie.*

Przykłady: *Czy dany graf można pokolorować trzema kolorami? Czy komiwojażer może obejść wszystkie miejscowości drogą o sumarycznej długości mniejszej niż 125 km?* W problemach optymalizacyjnych żądamy więcej, mianowicie rozwiązania podanego w postaci liczb, znaków, list itp. *Ile najmniej kolorów potrzeba, aby pokolorować dany graf? Któredy przebiega optymalna trasa komiwojażera i jaka jest jej długość?*

Problemy decyzyjne są bardziej podstawowe i prościej postawione niż optymalizacyjne. Co więcej, pewne problemy optymalizacyjne można sprowadzić do sekwencji problemów decyzyjnych. Weźmy przykład problemu komiwojażera. Niech najmniejsza różnica między odległościami miast w problemie komiwojażera wynosi przykładowo $\epsilon = 1$ km (liczbę tę nazwijmy rozdzielcością problemu), a długość drogi optymalnej 66 km. Aby znaleźć optymalną trasę, możemy zapytać „decyzyjnie”: *Czy istnieje trasa komiwojażera o długości mniejszej niż 100 km?* Jeśli odpowiedź jest twierdząca, to zapytamy dalej: *Czy istnieje trasa komiwojażera o długości mniejszej niż 50 km?* Jeśli odpowiedź jest przecząca, zadajemy kolejne pytanie, tym razem z liczbą 75 km itd. W ten sposób, stosując tę metodę *bisekcji*, czyli połowienia podziału, zawężamy przedział możliwej długości drogi komiwojażera aż do momentu, gdy jest on po siódmym podziale pomiędzy 65.6 i 66.4 km. Wiemy, że ta trasa jest optymalna, bo każda trasa krótsza musiałaby



Rysunek 6.8: Metoda bisekcji zmiany problemu optymalizacyjnego na problem decyzyjny. W kolejnych krokach (pionowe kreski) pytamy, czy istnieje trasa o długości mniejszej niż 100 km, następnie mniejszej niż 50 km, mniejszej niż 75 km itd. W ten sposób zawężamy przedział możliwej długości drogi komiwojażera aż do momentu, gdy jego długość jest mniejsza od rozdzielcości problemu ϵ .

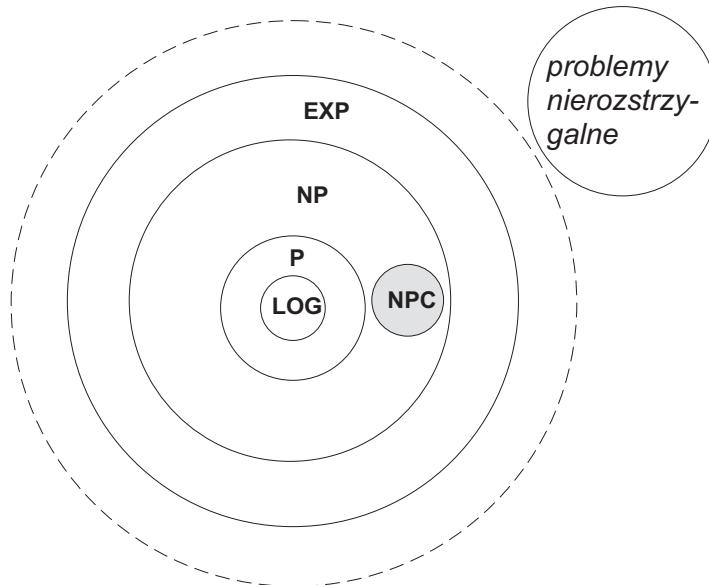
być krótsza o co najmniej $\epsilon = 1$ km, a byłoby to więcej niż długość znalezionej przedziału zwierającejgo rozwiązanie. Procedura bisekcji zobrazowana jest na rys. 6.8. Jeśli długość przedziału wynosi d , to liczba potrzebnych kroków bisekcji wynosi $N_{\text{bis}} = \lceil \log_2 \frac{d}{\epsilon} \rceil$. Dla d proporcjonalnego do długości danych n , jak w problemie komiwojażera, $N_{\text{bis}} \sim \log_2 n$, jest to więc bardzo wolno zmieniająca się funkcja.

Złożoność problemów decyzyjnych klasyfikujemy ze względu na zależność czasu t wykonywania obliczeń przez najlepszy możliwy algorytm od długości danych n . Podstawowe klasy, uszeregowane rosnąco ze stopniem złożoności obliczeniowej, są następujące (patrz rys. 6.9):

- logarytmiczna, $t = \mathcal{O}(\log n)$ – LOG,
- wielomianowa, $t = \mathcal{O}(n^m)$, gdzie m jest liczbą naturalną niezależną od n – P (od ang. *polynomial*),
- niedeterministyczna wielomianowa (problemy rozwiązywalne na *niedeterministycznej* maszynie Turinga w czasie wielomianowym, $t = \mathcal{O}(n^m)$) – NP,
- wykładnicza, $t = \mathcal{O}(\exp(an))$ ($a > 0$) – EXP.

Podział ten jest w pewnej mierze umowny, ponieważ istnieją problemy o jeszcze większej złożoności, np. $t = \mathcal{O}(e^{e^n})$, oraz o złożoności mniejszej, np. $\log \log n$, i nie ma tu ani górnego, ani dolnego ograniczenia. Mogą też oczywiście występować złożoności pośrednie, np. $n^2 \log n$ itd. Zauważmy, że zgodnie z definicją (1.4) symbolu \mathcal{O} zdefiniowane wyżej klasy zawierają się tak jak pokazano na rys. 6.9.

Na szczególną uwagę w powyższej liście zasługuje dyskutowana szczegółowo poniżej klasa NP. Do jej definicji służy niedeterministyczna maszyna Turinga



Rysunek 6.9: Klasyfikacja problemów decyzyjnych

z rys. 6.3, na której problemy tej klasy są rozwiązywalne w czasie wielomianowym.

Def. 6.12. *Problem klasy NP to problem rozwiązywalny na niedeterministycznej maszynie Turinga w czasie wielomianowym.*

Przykładem problemu klasy P jest problem liczb trójkątnych z rozdz. 1.3, a problem klasy EXP to np. pytanie *ile jest tras komiwojażera o długości mniejszej niż d* . Jest to problem klasy EXP, bo liczba takich tras może rosnąć wykładniczo z n . Inny problem klasy EXP, to problem Wież Hanoi (zmieniony na problem decyzyjny: *Czy możemy przełożyć wieże w czasie nie większym niż 2^n ?*). Aby wykonać algorytm Hanoi, potrzebujemy 2^n kroków, co więcej, nie jesteśmy w stanie tego procesu „zrównoleglić”, bo kolejne kroki występują po sobie sekwencyjnie – następny czeka na wynik poprzedniego. W tym przypadku wieloprocesorowość NDTM jest bezużyteczna. Na rys. 6.3 mamy w tym przypadku 2^n poziomów i wykonanie algorytmu na niedeterministycznej lub deterministycznej maszynie Turinga zajmuje 2^n kroków.

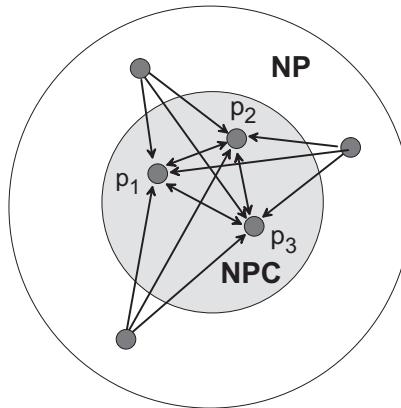
Wreszcie przykładem problemu klasy NP jest problem komiwojażera z podrozdziału 6.5. Przyjrzyjmy się bliżej, w jaki sposób niedeterministyczna maszyna Turinga rozwiązuje to zadanie. Spójrzmy na rys. 6.3. Zaczynamy z Pińczową z jedną maszyną. Mamy n możliwości dróg do n miejscowości, więc rozdzielamy maszynę na n kopii i każdą kopię posyłamy do innego miasta. W każdym z miast mamy teraz $n - 1$ możliwości wyboru dalszej drogi, klonujemy więc każdą z n maszyn na $n - 1$ kopii, w kolejnym mieście dokonujemy dalszego podziału na $n - 2$ kopii itd. W ostatnim kroku mamy $n!$ maszyn. Natomiast czas wykonywania algorytmu jest proporcjonalny do n . Jest tak dzięki równoległości obliczeń: w każdym kolejnym mieście mamy coraz więcej maszyn i nasza moc obliczeniowa rosła. A teraz bardzo istotna obserwacja: w przypadku problemu komiwojażera liczba poziomów na rys. 6.3 jest n , tyle, co liczba miast. Co za tym idzie, czas wykonywania całego algorytmu wynosi n i problem jest klasy NP. W ogólności, dla problemów klasy NP liczba poziomów na rys. 6.3 rośnie wielomianowo z n . Sformułujmy nasze spostrzeżenia w nieco inny sposób:

Wniosek 6.1. *Dla problemów klasy NP istnieje algorytm, dla którego drzewo możliwości wykonania rys. 6.3 ma wysokość rosnącą wielomianowo z n .*

Jeszcze inaczej:

Wniosek 6.2. *Problem jest klasy NP, jeśli możemy sprawdzić poprawność rozwiązania w czasie wielomianowym (choć samo rozwiązanie problemu może wymagać czasu dłuższego).*

Przykładowo, mając już z trudem uzyskane rozwiązanie dla decyzyjnego problemu komiwojażera (*czy istnieje trasa krótsza od d*), możemy je bardzo łatwo



Rysunek 6.10: Redukowalność wielomianowa problemów klasy NP do NPC: każdy problem NP jest redukowalny wielomianowo do problemu NPC

zweryfikować: wystarczy dodać n odległości wzdłuż trasy i porównać wynik z d . Liczba operacji oczywiście rośnie wielomianowo (w tym przypadku liniowo) z n .

Teraz dalsze definicje:

Def. 6.13. Mówimy, że problem p_1 jest redukowalny w czasie wielomianowym (w skrócie: P-redukowalny) do algorytmu p_2 (piszemy $p_1 \rightarrow p_2$), jeśli istnieje algorytm wielomianowy sprowadzający p_1 do p_2 .

Oznacza to, że A_1 możemy „przerobić” na A_2 , a algorytm przekształcający jest wielomianowy. W klasie NP na rys. 6.9 wyróżniamy podklasę NPC (ang. non-deterministic polynomial complete) tzw. problemów NP-zupełnych.

Def. 6.14. Problem p klasy NP jest zupełny, jeżeli każdy inny problem klasy NP jest P-redukowalny do p .

Tak więc każdy problem NP jest P -redukowalny do pewnego problemu NPC, a problemy NPC są wzajemnie do siebie redukowalne, zob. rys 6.10. Ponieważ relacja P -redukowalności jest przechodnia, $p_1 \rightarrow p_2 \rightarrow p_3 \Rightarrow p_1 \rightarrow p_3$, aby pokazać, że dany problem p jest klasy NPC wystarczy pokazać, że p jest P -redukowalny do pewnego znanego już problemu NPC oraz że pewien problem NPC jest redukowalny do P (patrz rys. 6.10).

A teraz jeden z najważniejszych nieroziwiążanych problemów informatyki teoretycznej.

Nieroziwiążany problem 6.1. Czy klasa NP jest różna od klasy P?

Innymi słowy, czy nie da się jednak rozwiązać problemów NP w czasie wielomianowym tak, jak potrafimy je zweryfikować w czasie wielomianowym? Jak

dotąd, nie udało się znaleźć ani twierdzącej, ani przeczącej odpowiedzi na podstawowe pytanie 6.1. Aby uzmysłowić sobie, że pytanie 6.1 nie jest bez sensu, rozważmy siostrzany problem do problemu istnienia cyklu Hamiltona w grafie, mianowicie problem istnienia cyklu Eulera. Gdybyśmy nie znali teorii grafów, moglibyśmy dla danego grafu o k krawędziach i n wierzchołkach sprawdzać wszystkie możliwe drogi, co prowadziłoby do złożoności obliczeniowej $(k - 1)!$. Jednak na mocy Tw. 5.2 wiemy, że wystarczy sprawdzić parzystość stopnia każdego wierzchołka, co oczywiście jest procesem wielomianowym. Tak więc wiedza teoretyczna pozwala nam znacznie zredukować czas rachunku. Problem 6.1 jest analogicznym pytaniem, dotyczącym problemów klasy NP: *Czy jest to tylko nasza niewiedza, że nie umiemy tych problemów rozwiązać wielomianowo, czy też jest to fundamentalna prawda matematyczna?* Większość informatyków sądzi, że odpowiedź na pytanie 6.1 jest twierdząca, tj. $\text{NP} \neq \text{P}$.

Szczególnego znaczenia nabierają tu problemy NPC. Gdyby udało się którykolwiek z nich rozwiązać w czasie wielomianowym, oznaczałoby to, że każdy z problemów NP byłby w istocie wielomianowy, czyli $\text{NP} = \text{P}$. Wśród ok. 1000 popularnych problemów klasy NP bardzo duże znaczenie praktyczne ma problem rozkładu liczby na czynniki pierwsze (który nie jest klasy NPC). Rozważmy liczbę n -cyfrową, która ma dwa czynniki pierwsze. Ich znalezienie wymaga czasu wykładniczego w n (nie znamy lepszego algorytmu), podczas gdy sprawdzenie rozwiązania jest wielomianowe, $\sim n^2$, bo musimy wykonać tylko proste mnożenie. Na tej asymetrii oparte jest szyfrowanie algorymem RSA, będące podstawą kriptografii z kluczem publicznym [60]. Gdyby więc ktoś znalazł algorytm rozwiązujący problem faktoryzacji w czasie wielomianowym¹¹, złamałby tym samym szyfr RSA i uniemożliwił przekazywanie poufnych danych do banku czy zakupy przez internet.

Na koniec podamy jeszcze dla porządku kilka często używanych definicji związanych z teorią złożoności problemów decyzyjnych.

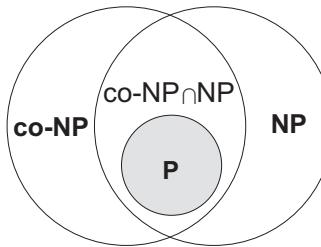
Def. 6.15. *Problemem dopełniającym do danego problemu decyzyjnego nazywamy problem, w którym pytanie jest zanegowane.*

Na przykład problemem dopełniającym do problemu „czy graf G ma cykl Hamiltona” jest problem „czy graf G nie ma cyklu Hamiltona”.

Def. 6.16. *Klasą co- X nazywamy klasę problemów dopełniających do problemów klasy X .*

Można pokazać, że dla klasy X algorytmów deterministycznych zachodzi równość $X = \text{co-}X$. Jest tak, ponieważ tym samym algorymem można odpowiedzieć na

¹¹ Właściwość tę mają algorytmy dla komputerów kwantowych [61–63]. Jest to jednym z głównych powodów, dla których prace nad konstrukcją komputerów kwantowych są bardzo intensywnie rozwijane.



Rysunek 6.11: Najbardziej spodziewana relacja między klasami co-NP, NP i P. Wszystkie obszary na diagramie oznaczają zbiory niepuste

pytanie „czy cecha c zachodzi dla przypadku p ”, jak też „czy c nie zachodzi dla p ”. Co ciekawe, dla klasy co-NP nie wiadomo, czy $\text{co-NP} = \text{NP}$.

Nierozwiążany problem 6.2. Czy $\text{co-NP} = \text{NP}$?

Aby zrozumieć, na czym polega trudność i co powoduje brak symetrii, weźmy ponownie przykład z szukaniem cyklu Hamiltona. Jeśli kolega nam zademonstruje przez wskazanie, że dany graf G ma cykl Hamiltona, łatwo, tj. w czasie wielomianowym, sprawdzimy, czy ma rację. Jeśli jednak powie, że G nie posiada cyklu Hamiltona, aby to stwierdzenie zweryfikować, musimy wykonać algorytm niedeterministyczny, tj. sprawdzić bardzo wiele możliwości. Weryfikacja jest więc w tym przypadku tak samo trudna jak znalezienie samego rozwiązania.

Mimo braku dowodów na podstawowe problemy teorii złożoności algorytmów 6.1 i 6.2, większość badaczy uważa, że sytuacja jest taka, jak przedstawiono na rys. 6.11, tzn. $\text{NP} \neq P$ oraz $\text{co-NP} \neq \text{NP}$.

Zamiast czasu obliczeń do klasyfikacji problemów można też użyć wielkości obszaru pamięci niezbędnej do wykonania zadania. Wprowadzamy wtedy oznaczenia LOG-SPACE, P-SPACE itd., podczas gdy klasyfikacja wg czasu obliczeń to LOG-TIME, P-TIME, itd. Ponieważ pisanie i czytanie z pamięci zajmuje pewien czas, czas obliczeń problemu X-SPACE ($X=\text{LOG}, \text{P}$ itd.) jest przynajmniej klasą X-TIME. W związku z tym klasa X-SPACE zawiera klasę X-TIME.

W ostatnich latach prowadzone są bardzo intensywne i przynoszące sukcesy badania nad komputerami kwantowymi [61–63]. W związku z tym wyróżnione zostały nowe możliwe klasy złożoności obliczeniowej. W szczególności klasa QP (Quantum Polynomial) zawiera algorytmy wykonywalne na komputerze kwantowym w czasie wielomianowym. Sądzi się, że zachodzi zawieranie $P \subseteq QP \subseteq \text{NP}$. Ponieważ jednak, jak wspomniano powyżej, nie ma dowodu że $\text{NP} \neq P$, nie wiemy też na pewno, czy klasa QP jest istotnie różna od P i NP. Te fascynujące zagadnienia są przedmiotem aktualnych dociekań w teorii algorytmów.

Ćwiczenia

- 6.1. Skonstruuj maszynkę sprawdzającą, czy dana liczba w zapisie dwójkowym jest podzielna przez 5.
- 6.2. Skonstruuj maszynkę sprawdzającą, czy dana liczba w zapisie dziesiętnym jest podzielna przez 3.
- 6.3. Skonstruuj maszynę Turinga wykonującą sumę dwóch liczb naturalnych.
- 6.4. Udowodnij, że problem znajdowania cyklu Hamiltona w grafie jest P -redukowalny do decyzyjnego problemu komiwojażera.
- 6.5. Wykaż, że optymalny cykl rozwiązania euklidesowego problemu komiwojażera na płaszczyźnie nie może mieć skrzyżowanych krawędzi.

Rozdział 7

Wskazówki i odpowiedzi do ćwiczeń

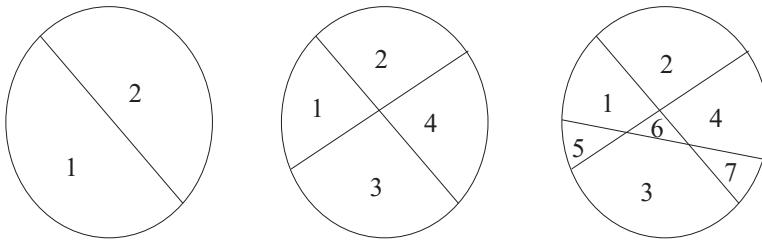
7.1 Rozdział 1

- Ćw. 1.1. Metoda równania charakterystycznego daje $a_n = \frac{1}{3} [2^n + (-1)^{n+1}]$. Funkcja tworząca ma postać

$$f(z) = \frac{z}{1 - z - 2z^2} = \frac{1}{3} \left[\frac{1}{1 - 2z} - \frac{1}{1 + z} \right] = \sum_{n=1}^{\infty} a_n z^n.$$

Pierwszych kilka wyrazów: $(a_n) = (1, 1, 3, 5, 11, 21, 43, 85, 171, 341, \dots)$.

- Ćw. 1.2. W oczywisty sposób mamy $p_1 = 2$, $p_2 = 4$. Trzecią prostą możemy przeprowadzić na trzy sposoby: 1) równolegle do którejś z istniejących prostych, powstaje wtedy łącznie 6 obszarów; 2) przecinając punkt przecięcia istniejących prostych, wtedy również uzyskujemy 6 obszarów; wreszcie 3) przecinając istniejące proste w innych punktach niż ich punkt przecięcia – wówczas otrzymujemy łącznie 7 obszarów. Oczywiście metoda 3) jest najlepsza. Przecięliśmy dwie proste i powstały 3 nowe obszary. Kontynuując te rozważania dla większej liczby prostych, widzimy, że najwięcej nowych



Rysunek 7.1: Dzielenie pizzy prostoliniowymi cięciami noża

obszarów uzyskamy, prowadząc cięcie numer n w taki sposób, aby przeciąć istniejące $n - 1$ prostych w punktach, które nie są dotychczasowymi punktami przecięcia tych prostych. Zawsze można tak zrobić, wystarczy wybrać prostą, która nie jest równoległa do żadnej z istniejących prostych. Jeśli przypadkowo natrafimy na istniejący punkt przecięcia, to przesuwamy nieco naszą prostą. Robiąc kilka rysunków (rys. 7.1), zauważamy, że nasze cięcie zwiększa liczbę obszarów o n .

Dowód opiera się na następującym rozumowaniu. Nowa linia o numerze n przecina istniejące $n - 1$ linii w $n - 1$ punktach, przez które dzielona jest na n odcinków. Na przykład linia dodana na prawej części rysunku 7.1, o $n = 3$, jest podzielona na 3 odcinki. Po obu stronach każdego z tych odcinków znajduje się „stary” obszar oraz obszar nowy. Tak więc podążając wzduż linii od jej lewego końca w prawo, mamy z jednej strony stary obszar o numerze 1, a z drugiej nowy obszar o numerze 5. Dalej, przy drugim odcinku obszary 3 i 6 oraz przy trzecim odcinku obszary 4 i 7. Mamy więc 3 nowe obszary: 5, 6 i 7. Dla linii o numerze n , podzielonej na n odcinków, dostajemy n nowych obszarów. Rekurencja ma zatem postać analogiczną do liczb trójkątnych, ale z innym warunkiem początkowym:

$$p_n = p_{n-1} + n, \quad p_1 = 2.$$

Rozwiązanie ma postać $p_n = n(n + 1)/2 + 1$.

- Ćw. 1.3. Napiszmy $L_n = aL_{n-1} + bL_{n-2}$ i znajdźmy stałe a i b . Podstawiając do tego równania $L_k = F_k + 2F_{k-1}$ i wykorzystując rekurencję Fibonacciego $F_k = F_{k-1} + F_{k-2}$, otrzymujemy $a = b = 1$, czyli

$$L_n = L_{n-1} + L_{n-2}.$$

Równanie rekurencyjne ma zatem tę samą postać, co dla ciągu Fibonacciego, natomiast warunki początkowe są inne: $L_2 = 3$, $L_3 = 4$. Przedłużając rekurencję „do tyłu” za pomocą $L_n = L_{n+2} - L_{n+1}$, dostajemy $L_1 = 1$. Metoda równania charakterystycznego daje $L_n = \varphi^n + \hat{\varphi}^n$. Kilka pierwszych wyrazów to $(L_n) = (1, 3, 4, 7, 11, 18, 29, 47, 76, 123, \dots)$. Inną metodą jest zastosowanie funkcji tworzącej. Wprowadźmy $g(z) = \sum_{n=1}^{\infty} L_n z^n$,

a $f(z) = z/(1 - z - z^2)$ niech oznacza funkcję tworzącą dla ciągu Fibonacciego. Otrzymujemy $g(z) - L_1z = f(z) - F_1z + 2zF(z)$, skąd

$$g(z) = \frac{(1 + 2z)z}{1 - z - z^2}.$$

Rozkład na ułamki proste i rozwinięcie szeregu geometrycznego daje ponownie powyższy wzór ogólny.

- Ćw. 1.4. Związek wynika w prosty sposób z ogólnej postaci F_n i L_n .
- Ćw. 1.5. Związki (a) i (b) można elementarnie wyprowadzić, korzystając z wzoru na sumę częściową szeregu geometrycznego,

$$\sum_{i=0}^n a^i = \frac{1 - a^{n+1}}{1 - a}.$$

Różniczkując to wyrażenie po a i mnożąc wynik przez a , dostajemy wzór

$$\sum_{i=0}^n i a^i = \frac{a(-(n+1)a^n + na^{n+1} + 1)}{(1-a)^2},$$

potrzebny do wyprowadzenia wzoru (c). Dalej postępowanie jest następujące: rozważamy wzór ogólny dla liczb Fibonacciego do sum i korzystamy z powyższych wzorów dla $a = \phi$ i $a = \hat{\phi}$. Następnie upraszczamy wyrażenia za pomocą związków

$$\phi + \hat{\phi} = 1, \phi\hat{\phi} = -1, (1 - \phi)(1 - \hat{\phi}) = -1, 1 - \phi^2 = -\phi, 1 - \hat{\phi}^2 = -\hat{\phi}.$$

Związek (d) można w prosty sposób udowodnić indukcyjnie. Twierdzenie zachodzi dla $n = 1$, bo $F_1 = F_3 - 1$. Dodając stronami (d) oraz (b) dla n zwiększonego o 1,

$$F_1 + F_3 + F_5 + \cdots + F_{2n+1} = F_{2n+2},$$

otrzymujemy

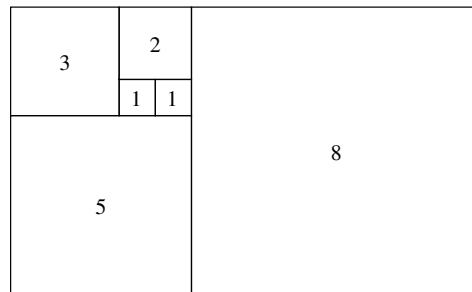
$$\begin{aligned} F_{2n+1} + 2F_{2n-1} + 3F_{2n-3} + 4F_{2n-5} + \cdots + (n-1)F_3 + nF_1 \\ = F_{2n+2} + F_{2n+1} - 1. \end{aligned}$$

Z definicji ciągu Fibonacciego przekształcamy prawa stronę,

$$F_{2n+1} + 2F_{2n-1} + 3F_{2n-3} + 4F_{2n-5} + \cdots + (n-1)F_3 + nF_1 = F_{2n+3} - 1,$$

co jest tezą indukcyjną dla $n + 1$.

Związek (e) można udowodnić geometrycznie. Suma pól n kwadratów o długości boków F_i , ułożonych jak na rys. 7.2, wynosi $F_n F_{n+1}$.



Rysunek 7.2: Rysunek ukazujący, że $\sum_{i=1}^n F_i^2 = F_n F_{n+1}$. Etykiety oznaczają długości boków

- Ćw. 1.6. Stosując iteracyjnie rekurencję Fibonacciego do pierwszych wyrazów po znaku równości, otrzymujemy

$$\begin{aligned} F_n &= F_{n-1} + F_{n-2} = 2F_{n-2} + F_{n-3} = 3F_{n-3} + 2F_{n-4} \\ &= 5F_{n-4} + 3F_{n-5} = 8F_{n-5} + 5F_{n-6} = 13F_{n-6} + 8F_{n-7} \\ &= \dots = F_{i+1}F_{n-i} + F_iF_{n-i-1}. \end{aligned}$$

- Ćw. 1.7 Oznaczając ułamek jako u , zauważamy, że

$$u = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\dots}}}}$$

skąd $u^2 - u - 1 = 0$, co daje $u = \phi$ lub $u = \hat{\phi}$. Ponieważ u jest dodatnie, pozostaje możliwość $u = \phi$.

- Ćw. 1.8. $a_n = -2 + 3 \cdot 2^n - 3^n$.
- Ćw. 1.9. $a_n = \frac{1}{18} (6n(-1)^n - 14(-1)^n + 5 \cdot 2^n)$.
- Ćw. 1.10. $u_n = 6 \cdot 2^n - n^2 - 4n - 6$, $(u_n) = (1, 6, 21, 58, 141, 318, \dots)$. Przeniesiona masa to u_n razy masa najmniejszego krążka.
- Ćw. 1.11. Jedna płaszczyzna dzieli przestrzeń na dwa obszary, więc $a_1 = 2$. Wyobraźmy sobie, że mamy n płaszczyzn i dodajemy kolejną o numerze $n+1$. Przecięcie każdych dwóch płaszczyzn tworzy prostą. Przecięcie nowej płaszczyzny ze starymi utworzy sieć prostych wyglądających po zrzutowaniu jak na rys. 7.1. W szczególności dodanie trzeciej płaszczyzny do dwóch istniejących prowadzi do 3 prostych jak na prawej części rys. 7.1. Proste te dzielą nową płaszczyznę na $p_n = n(n+1)/2 + 1$ obszarów (problem dzielenia

pizzy). Powtarzając w trzech wymiarach rozumowanie z zad. 1.2, wnioskujemy, że powstaje p_n nowych obszarów w przestrzeni, np. stary pozostaje na spodzie nowej płaszczyzny, a nowy tworzony jest na górze. Stąd rekurencja

$$a_{n+1} = a_n + p_n.$$

Stosując metodę rozwiązywania rekurencji niejednorodnej z rozdz. 1.6, otrzymujemy

$$a_n = \frac{n^3 + 5n + 6}{6}.$$

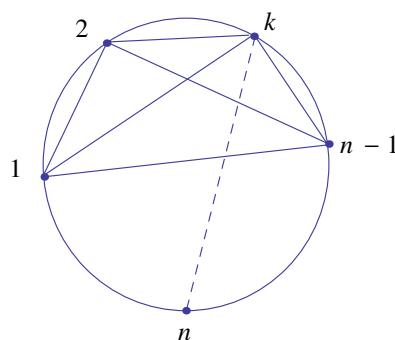
Pierwszych kilka wyrazów to

$$2, 4, 8, 15, 26, 42, 64, 93, 130, 176, \dots$$

- Ćw. 1.12. Działając na obie strony logarytmem przy podstawie 2 i oznaczając $b_n = \log_2 a_n$, otrzymujemy rekurencję liniową $b_{n+1} = 2b_n$, $b_1 = 1$. Daje to $b_n = 2^{n-1}$ i

$$a_n = 2^{2^{n-1}}.$$

- Ćw. 1.13. Zadanie to, pochodzące z [64], można rozwiązać indukcyjnie. Oznaczmy liczbę obszarów jako t_n . Dla $n = 2$ (jednej cięciwy) mamy dwa obszary, $t_2 = 2$. Rozważmy sytuację, gdy do $n - 1$ punktów dodajemy n -ty, co jest zobrazowane na rys. 7.3 dla $n = 5$. Wyobraźmy sobie, że idziemy nową cięciwą (linia przerywana) do góry od punktu n do punktu k . W momencie napotkania idącej w poprzek „starej” cięciwy (linia ciągła) obszar, przez który szliśmy, zostaje podzielony na dwa, tj. mamy o jeden obszar więcej. Idąc wzduż linii przerywanej przecinamy łącznie $(k-1)(n-k-1)$ starych cięciw – jest to iloczyn liczby „starych” punktów znajdujących się na lewo od punktu k i na prawo od punktu k . Przy dojściu do punktu k też dzielimy obszar, łącznie uzyskujemy więc $(k-1)(n-k-1) + 1$ obszarów



Rysunek 7.3: Ilustracja dowodu z ćw. 1.13

przy dodaniu cięciwy między punktami n i k . Musimy jeszcze zsumować po nowych cięciwach po k od 1 do $(n - 1)$, a więc

$$t_n = t_{n-1} + \sum_{k=1}^{n-1} [(k-1)(n-k-1) + 1].$$

Sumę możemy wykonać stosując wzory

$$\begin{aligned}\sum_{k=1}^i k &= i(i+1)/2, \\ \sum_{k=1}^i k^2 &= i(i+1)(1+2i)/6,\end{aligned}$$

co daje rekurencję niejednorodną

$$t_n = t_{n-1} + (n^3 - 6n^2 + 17n - 12)/6.$$

Rozwiązań otrzymujemy z pomocą Tw. 1.7, znajdując ostatecznie

$$t_n = (n^4 - 6n^3 + 23n^2 - 18n + 24)/24.$$

Pierwsze wyrazy, począwszy od $n = 1$, to 1, 2, 4, 8, 16, 31, 57, 90, ... Zwodniczość tego ciągu polega na tym, że pierwszych pięć wyrazów jest kolejnymi potęgami dwójki 2^{n-1} , ale dalsze coraz bardziej się od tego wzoru oddalają.

Inne rozwiązanie problemu podane jest szczegółowo w [6]. W wyniku otrzymuje się następującą sumę symboli Newtona,

$$\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \binom{n-1}{3} + \binom{n-1}{4},$$

równą rozwiązaniu 7.1.

- Ćw. 1.14. Nasz truteń ma tylko matkę-królową (nie ma tatusia!), babcię i dziadka ze strony matki, prababcię i pradziadka ze strony babci oraz prababcię ze strony dziadka itd. W pokoleniu n (licząc naszego trutnia za pierwsze pokolenie) znajduje się K_n królowych i T_n trutni. Ze schematu rozmnażania wynika, że $K_n = K_{n-1} + T_{n-1}$ (każdy truteń i królowa mają swoją matkę) oraz $T_n = K_{n-1}$ (każda królowa ma ojca). Stąd $K_n = K_{n-1} + K_{n-2}$, $K_1 = 0$, $K_2 = 1$, a więc $K_n = F_{n-1}$, a $T_n = K_{n-1} = F_{n-2}$. Licząc obie płcie, mamy $K_n + T_n = F_{n-1} + F_{n-2} = F_n$. Liczby osobników w danym pokoleniu tworzą ciąg Fibonacciego!
- Ćw. 1.15. Model zakłada, że dorosła para płodzi w jednym okresie rozrodczym dokładnie k par. Niech D_n oznacza liczbę dojrzałych par w okresie n , M_n liczbę młodych par, a $K_n = D_n + M_n$ całkowitą liczbę par. Mamy zatem

rekurencję $D_n = D_{n-1} + M_{n-1}$, $M_n = kD_{n-1}$, co daje $D_n = D_{n-1} + kD_{n-2}$, $M_n = M_{n-1} + kM_{n-2}$ oraz $K_n = K_{n-1} + kK_{n-2}$ z warunkami początkowymi $K_1 = K_2 = 1$. Rozwiąza niem jest

$$K_n = \frac{2^{-n} \left((\sqrt{4k+1} + 1)^n - (1 - \sqrt{4k+1})^n \right)}{\sqrt{4k+1}}.$$

Oczywiście dla przypadku $k = 1$ otrzymujemy ciąg Fibonacciego.

- Ćw. 1.16. Oznaczmy liczby par królików w okresie n w następujący sposób: D_n – dorosłe, $M_n^{(1)}$ – małe w wieku 1, $M_n^{(2)}$ – małe w wieku 2, …, $M_n^{(m)}$ – małe w wieku m , K_n – całkowita liczba par. Zachodzą następujące związki:

$$\begin{aligned} D_n &= D_{n-1} + M_{n-1}^{(m)}, \\ M_n^{(m)} &= M_{n-1}^{(m-1)}, \\ M_n^{(m-1)} &= M_{n-2}^{(m-2)}, \\ &\dots \\ M_n^{(2)} &= M_{n-1}^{(1)}, \\ M_n^{(1)} &= D_{n-1}. \end{aligned}$$

Pierwszy związek oznacza, że najstarsze króliki wydoroślały, kolejne związki, że małe dojrzewają – w każdym okresie starzeją się o 1, a ostatni związek, że dorosłe płodzą małe. Wstawiając drugi związek do pierwszego, potem trzeci itd., otrzymujemy rekurencję $D_n = D_{n-1} + D_{n-m-1}$. Ze związków $M_n^{(j)} = D_{n-j}$, $j = 1, \dots, m$, wynika, że również $M_n^{(j)} = M_{n-1}^{(j)} + M_{n-m-1}^{(j)}$ oraz ostatecznie $K_n = K_{n-1} + K_{n-m-1}$. Równanie charakterystyczne ma postać

$$x^{m+1} = x^m + 1.$$

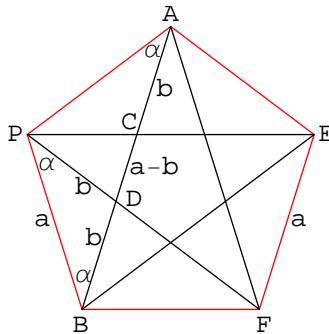
Dla $m = 1$, oczywiście, otrzymujemy przypadek Fibonacciego. Dla $m = 2$ lub $m = 3$ równanie charakterystyczne daje się rozwiązać, ale wzory są bardzo długie. Dla $m > 3$ można tylko postępować numerycznie, metoda przestaje więc być praktyczna. Oczywiście, im większe m , tym wolniejszy jest wzrost populacji.

- Ćw. 1.18. Dowód zilustrowany jest na rys. 7.4. Oznaczmy długość boku pentagramu przez a . Mamy $EF \parallel AD$ i $AE \parallel DF$, w związku z tym $|AD| = a$. Oznaczmy $|AC| = |BD| = b$. Wtedy $|AB| = a + b$. Z podobieństwa trójkątów PAB i PDB dostajemy

$$\frac{b}{a} = \frac{a}{a+b},$$

co jest antyczną definicją Złotego Podziału. Podobnie, z podobieństwa trójkątów PCD i PEF wynika, że

$$\frac{a-b}{b} = \frac{a}{a+b} = \frac{b}{a}.$$

Rysunek 7.4: Rysunek do dowodu, że $|AB|/|AD| = \phi = |AD|/|AC|$

- Ćw. 1.21. Rozwiążujemy rekurencję $a_k = pa_{k+1} + qa_{k-1}$ z warunkami $a_0 = 0$, $a_{150} = 1$ dla Krzysztofa i $a_{50} = 1$, $a_{200} = 0$ dla Ludwika. Niech $r = q/p$. Dla Krzysztofa

$$a_k = \frac{1 - r^k}{1 - r^{150}},$$

więc prawdopodobieństwo wygranej wynosi $a_{100} = 13\%$. Dla Ludwika (dla odróżnienia wprowadzamy prim)

$$a'(k) = \frac{r^k - r^{200}}{r^{50} - r^{200}},$$

zatem prawdopodobieństwo wygranej wynosi $1 - a'(100) = 1.6\%$. Strategia Krzysztofa jest zdecydowanie lepsza. Aby określić oczekiwany czas gry, rozwiązujemy rekurencję $t(k) = 1 + pt_{k+1} + qt_{k-1}$ z warunkami z warunkami $t_0 = 0$, $t_{150} = 0$ dla Krzysztofa i $t_{50} = 0$, $t_{200} = 0$ dla Ludwika. Wyniki wynoszą odpowiednio

$$t(k) = \frac{-kr^{150} + k + 150(r^k - 1)}{(p - q)(r^{150} - 1)}, \quad t(100) \simeq 4001$$

oraz

$$t'(k) = \frac{150r^k - (k - 50)r^{200} + (k - 200)r^{50}}{(p - q)r^{50}(r^{150} - 1)}, \quad t'(100) \simeq 2381.$$

Krzysztof będzie więc (średnio) grać niemal dwa razy dłużej od Ludwika. Znak tego efektu łatwo zrozumieć, ponieważ dla $p < 1/2$ stan posiadania gracza „dryfuje” w dół (zob. rys. 1.9). Ponieważ Krzysztof ma dolną granicę 0, a Ludwik 50, Krzysztof gra dłużej.

- Ćw. 1.22. Niech początkowy stan posiadania gracza wynosi k monet, a n oznacza maksymalną możliwą liczbę monet w jego posiadaniu (rozbicie banku). Po pierwszym rozdaniu gracz ma $k+1$ monet z prawdopodobieństwem p , lub $k-1$ monet z prawdopodobieństwem q . W pierwszym przypadku rozwiązujemy problem ruinę gracza startującego ze stanu $k+1$ z warunkami

$a(k) = 1$ (powrócił do k), $a(n) = 0$ (rozbił bank, więc nie powróci do k). W wyniku rozwiązania prawdopodobieństwo ponownego osiągnięcia stanu k monet wynosi

$$p \frac{r^{k+1} - r^n}{r^k - r^n}, \quad r = \frac{q}{p}.$$

W drugim przypadku, postępując analogicznie, otrzymujemy

$$q \frac{r^{k-1} - r^0}{r^k - r^0},$$

co łącznie daje prawdopodobieństwo całkowite

$$P = p \frac{r^{k+1} - r^n}{r^k - r^n} + q \frac{r^{k-1} - r^0}{r^k - r^0}.$$

Dla $p = 0.49$, $n = 200$ i $k = 100$ mamy $P = 98\%$, czyli niemal pewność!

- 1.23. Zadanie jest modyfikacją problemu ruiny gracza dla gry sprawiedliwej ($p = q = 1/2$), gdzie $a_0 = 1$, $a_n = 0$, oraz $n \rightarrow \infty$ (ulica Sienkiewicza jest nieskończona w jednym kierunku!). Rozwiązanie dla prawdopodobieństwa dojścia na dworzec i dla średniego czasu dojścia przy ustalonym n wynosi

$$a_k = \frac{n - k}{n}, \quad t_n = k(n - k).$$

Wynik ten możemy uzyskać z ogólnej metody lub też błyskawicznie z wzorów (1.46, 1.49) przez zastąpienie $k \rightarrow (n - k)$, co jest odległością od stanu pochłaniającego o prawdopodobieństwie 0. Biorąc następnie granicę $n \rightarrow \infty$, uzyskujemy

$$a_k = 1, \quad t_n \rightarrow \infty.$$

Tak więc profesor z pewnością dojdzie na dworzec, ale średnio zajmie mu to nieskończenie długi czas.

Uwaga: Technika zastosowana w rozwiązaniu, polegająca na zastąpieniu problemu z nieskończoną liczbą stanów przez problem o skończonym n , a następnie dokonanie przejścia granicznego $n \rightarrow \infty$, jest bardzo użyteczna i często stosowana.

- Ćw. 1.24. Niech liczby p i q mają po $3m$ cyfr, jeśli nie, to uzupełniamy je z przodu zerami. Piszymy $p = 10^{2m}l_2 + 10^ml_1 + l_0$ oraz $q = 10^{2m}r_2 + 10^mr_1 + r_0$. Pojawia się a priori 9 iloczynów $l_i r_j$ do wykonania. Można jednak użyć następującej sprytnej metody Tooma i Cooka. Tworzymy pomocnicze wielomiany

$$P(x) = x^2l_2 + xl_1 + l_0, \quad Q(x) = x^2r_2 + xr_1 + r_0.$$

Następnie wybieramy pewien dowolny zbiór pięciu niewielkich liczb całkowitych, np. $\{-2, -1, 0, 1, 2\}$ i obliczamy $W(x) = P(x)Q(x)$ dla tych wartości:

$$\begin{aligned} W(-2) &= (4l_2 - 2l_1 + l_0)(4r_2 - 2r_1 + r_0), \\ W(-1) &= (l_2 - l_1 + l_0)(r_2 - r_1 + r_0), \\ W(0) &= l_0 r_0, \\ W(1) &= (l_2 + l_1 + l_0)(r_2 + r_1 + r_0), \\ W(2) &= (4l_2 + 2l_1 + l_0)(4r_2 + 2r_1 + r_0). \end{aligned}$$

Jak dotąd wykonaliśmy 5 mnożeń liczb trzy razy krótszych od wyjściowych. Następnie zauważamy, że $W(x) = w_4x^4 + w_3x^3 + w_2x^2 + w_1x + w_0$. Układamy układ równań

$$\begin{aligned} 16w_4 - 8w_3 + 4w_2 - 2w_1 + w_0 &= W(-2), \\ w_4 - w_3 + w_2 - w_1 + w_0 &= W(-1), \\ w_0 &= W(0), \\ w_4 + w_3 + w_2 + w_1 + w_0 &= W(1), \\ 16w_4 + 8w_3 + 4w_2 + 2w_1 + w_0 &= W(2). \end{aligned}$$

Układ ten jest łatwy do rozwiązań, co można zrobić w czasie $\sim n$, otrzymując

$$\begin{aligned} w_0 &= W(0), \\ w_1 &= \frac{1}{42}(6W(-2) - 38W(-1) + 15W(0) + 18W(1) - W(2)), \\ w_2 &= \frac{1}{14}(-W(-2) + 11W(-1) - 20W(0) + 11W(1) - W(2)), \\ w_3 &= \frac{1}{42}(-6W(-2) + 17W(-1) - 15W(0) + 3W(1) + W(2)), \\ w_4 &= \frac{1}{14}(W(-2) - 4W(-1) + 6W(0) - 4W(1) + W(2)). \end{aligned}$$

Z konstrukcji wynika, że $W(10^m) = pq$. Obliczamy więc

$$pq = W(10^m) = 10^{4m}w_4 + 10^{3m}w_3 + 10^{2m}w_2 + 10^mw_1 + w_0,$$

co wymaga rzędu $\mathcal{O}(n)$ kroków. W ten sposób osiągnieliśmy cel. Rekurencja ma postać $T(n) = 5T(n/3) + \beta n$, więc na mocy Tw. o rekurencji uniwersalnej

$$T(n) = \Theta(n^{\log_3 5}) \simeq \Theta(n^{1.465}).$$

Uogólnienie dla podziału na s części wymaga $(2s - 1)$ mnożeń liczb s razy krótszych, więc $T(n) = (2s - 1)T(n/s) + \gamma n$, zatem $T(n) = \Theta(n^{\log_s(2s-1)})$.

- Ćw. 1.25. Stosując Tw. Akry-Bazziego, otrzymujemy równanie na p

$$\left(\frac{1}{2}\right)^{2p-1} + \left(\frac{1}{2}\right)^{p+1} = 1,$$

co jest równaniem kwadratowym na zmienną $y = (\frac{1}{2})^p$ o rozwiązaniu $p_0 = -2 + \log_2(1 + \sqrt{33}) \sim 0.754$. Następnie obliczamy

$$T(x) = \Theta\left(x^{p_0} \left[1 + \int_{x_0}^x u^{\alpha-1-p_0} du\right]\right) = \begin{cases} \Theta(x^{p_0}) & \text{dla } \alpha < p_0 \\ \Theta(x^{p_0} \log x) & \text{dla } \alpha = p_0 \\ \Theta(x^\alpha) & \text{dla } \alpha > p_0 \end{cases}.$$

- Ćw. 1.26. $T(n) = \Theta(\log n)$.
- Ćw. 1.27. Czas scalania jest proporcjonalny do sumy długości list L_1 i L_2 , więc rekurencja ma postać

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n),$$

co na mocy uogólnionego Tw. Akry-Bazziego daje $T(n) = \Theta(n \log n)$.

7.2 Rozdział 2

- Ćw. 2.1. Postępujemy indukcyjnie i korzystamy z całkowania przez części:

$$\begin{aligned} (n+1)! &= \int_0^\infty t^{n+1} e^{-t} dt = \int_0^\infty t^{n+1} (-e^{-t})' dt = \\ &-t^{n+1} e^{-t} \Big|_0^\infty + \int_0^\infty (t^{n+1})' e^{-t} dt = (n+1) \int_0^\infty t^n e^{-t} dt = (n+1)n!. \end{aligned}$$

- Ćw. 2.2. Zapisujemy

$$n!^2 = (1 \cdot 2 \cdot 3 \dots n)(n \dots 3 \cdot 2 \cdot 1) = 1 \cdot n \cdot 2 \cdot (n-1) \dots n \cdot 1 = \prod_{i=1}^n i(n+1-i).$$

Dla ustalonego n wyrażenie $i(n+1-i)$ możemy traktować jako funkcję kwadratową w zmiennej i , $1 \leq i \leq n$. Funkcja ta jest wypukła ku górze, więc przyjmuje minima na krańcach przedziału zmiennej i , tj. dla $i = 1$ lub $i = n$, obydwa o wartości n . Tak więc $n!^2 \geq \prod_{i=1}^n n = n^n$. Maksimum przypada dla $i = \frac{n+1}{2}$ jeśli n jest nieparzyste, wtedy wartość funkcji w maksimum wynosi $(n+1)^2/4$, oraz dla $i = \frac{n}{2}$ lub $i = \frac{n+2}{2}$ jeśli n jest parzyste, wówczas wartość w maksimum wynosi $n(n+1)/4 < (n+1)^2/4$. W związku z tym mamy ograniczenie $i(n+1-i) \leq \frac{1}{4}(n+1)^2$, czyli $n!^2 \leq \prod_{i=1}^n \frac{1}{4}(n+1)^2 = (\frac{n+1}{2})^{2n}$. Pierwiastkując uzyskane nierówności, otrzymujemy

$$n^{n/2} \leq n! \leq \left(\frac{n+1}{2}\right)^n.$$

Oszacowanie to nie jest wystarczająco dokładne, by znajdować praktyczne zastosowania.

- Ćw. 2.3. Dla n parzystych korzystamy ze wzoru $n!! = 2^n(n/2)!$ i wzoru Stirlinga (2.7). Dla n nieparzystych używamy związku $n!! = n!/(n-1)!!$.
- Ćw. 2.4. (a) $n(n-1)^{k-1}$, (b) $n^{k-1}(n-1)$.
- Ćw. 2.5. Każdy z n elementów zbioru A należy do zbioru X , zbioru Y lub do żadnego z nich. W związku z tym liczba możliwości jest równa liczbie n -elementowych wariacji z powtórzeniami 3-elementowego zbioru, tj. 3^n . Na przykład dla $A = \{a, b\}$ mamy 9 możliwości

$$(X, Y) = (\emptyset, \emptyset), (\{a\}, \emptyset), (\emptyset, \{a\}), (\{b\}, \emptyset), (\emptyset, \{b\}), (\{a, b\}, \emptyset), (\emptyset, \{a, b\}), (\{a\}, \{b\}), (\{b\}, \{a\}).$$

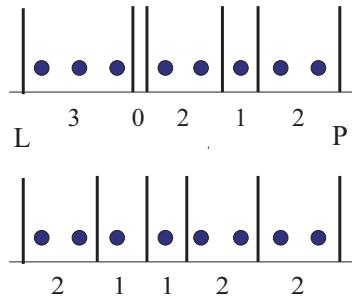
- Ćw. 2.6. Jeden punkt przecięcia powstaje dla każdej czwórki wierzchołków wielokąta, zatem całkowita liczba punktów przecięcia przekątnych dla wielokąta o n wierzchołkach wynosi C_4^n .
- Ćw. 2.7. Założymy, że ułożyliśmy n nieroróżnicjalnych kul w rzędzie i następnie ustaviamy $k+1$ tekturowych przegród w taki sposób, aby powstały pudła (zob. rys. 7.5). Dwie przegrody muszą być zawsze na krańcach, a pozostałe $k-1$ „przegrody wewnętrzne” gdzieś pomiędzy nimi. Te $k-1$ przegród też jest nieroróżnicjalnych, bo nie jest istotne, którą przegrodą rozdzielimy pudła. Powstałe z tej konstrukcji pudła mogą zawierać kule lub pozostać puste. Teraz zastosujemy bardzo częstą sztuczkę, mianowicie potraktujemy na chwilę zarówno kule, jak i $k-1$ przegród wewnętrznych jako rozróżnialne. Każda z sytuacji z rys. 7.5 odpowiada teraz pewnej permutacji $n+k-1$ obiektów (kule + przegrody wewnętrzne), co możemy oczywiście zrobić na $(n+k-1)!$ sposobów. Następnie „poprawiamy” ten wynik: ponieważ kule są nieroróżnicjalne, policzyliśmy niepotrzebnie aż $n!$ ustawień kul w rzędzie, podczas gdy jest tylko jedno. Musimy więc podzielić wynik przez $n!$. Podobnie, przegrody wewnętrzne, których jest $k-1$, są rozróżnialne, więc dzielimy przez $(k-1)!$. Ostatecznie, rozwiązanie ma postać

$$\frac{(n+k-1)!}{n!(k-1)} = \binom{n+k-1}{n}.$$

Zauważmy jeszcze, że w wyniku naszej procedury pudła pozostają rozróżnialne, ponieważ dwie skrajne przegrody, lewa i prawa, są rozróżnialne. Tak więc możemy je ponumerować np. wg odległości od lewej ściany.

- Ćw. 2.8. Ćwiczenie to jest w istocie tym samym zadaniem, co ćw. 2.7. Należy po prostu utożsamić rodzaj kuli z numerem pudła, w którym się znajduje. Na przykład dla górnej części rys. 7.5 mamy 3 kule pierwszego rodzaju, 0 drugiego, 2 trzeciego, 1 czwartego i 2 piątego rodzaju. Tak więc liczba wyboru n obiektów z powtórzeniami spośród k obiektów (kombinacje z powtórzeniami) wynosi

$$\bar{C}_k^n = \binom{n+k-1}{n}.$$



Rysunek 7.5: Dwa przykłady rozmieszczenia $n = 8$ nieroróżniczalnych kul w $k = 5$ rozróżnialnych pudłach. Pionowe kreski to przegrody tworzące pudła. L i P oznaczają skrajne przegrody, które są rozróżnialne. Liczby poniżej rysunków podają liczbę kul w poszczególnych pudłach

- Ćw. 2.9. Możliwe są konfiguracje, gdzie dokładnie 1, 2 lub 3 pary mają taki sam rodzaj ciasta. W pierwszej sytuacji możemy wybrać parę o tym samym cieście na $C_1^5 = 5$ sposobów, a pozostałe 4 osoby biorą po jednym kawałku z każdego z pozostałych rodzajów ciasta. Kawałki ciasta możemy spermutować na $6!$ sposobów. Ponieważ kawałki wybrane przez parę są nieroróżniczalne (są tego samego rodzaju), dzielimy przez $2!$. Daje to

$$n_1 = 5 \frac{6!}{2!} = 1800.$$

W drugim przypadku możemy wybrać dwie pary o tym samym cieście na $C_2^5 = 10$ sposobów, pozostałe 2 osoby mogą wygrać po jednym rodzaju z trzech, co daje $C_2^3 = 3$ możliwości. Kawałki ciasta możemy spermutować na $6!$ sposobów. Ponieważ kawałki wybrane przez każdą z dwóch par są nieroróżniczalne, dzielimy przez $2!^2$. Zatem

$$n_2 = 10 \cdot 3 \frac{6!}{2!^2} = 5400.$$

W trzecim przypadku możemy wybrać trzy pary o tym samym cieście na $C_3^5 = 10$ sposobów. Kawałki ciasta możemy spermutować na $6!$ sposobów. Ponieważ kawałki wybrane przez każdą z trzech par są nieroróżniczalne, dzielimy przez $2!^3$. Zatem

$$n_3 = 10 \frac{6!}{2!^3} = 900.$$

Łącznie $n_1 + n_2 + n_3 = 8100$.

- Ćw. 2.10. Funkcja tworząca dla współczynników dwumianowych ma postać (zob. 2.2)

$$f(z) = \sum_{k=0}^n z^k \binom{n}{k} = (1+z)^n.$$

Różniczkując po z , otrzymujemy

$$\frac{df(z)}{dz} = \sum_{k=0}^n kz^{k-1} \binom{n}{k} = n(1+z)^{n-1}.$$

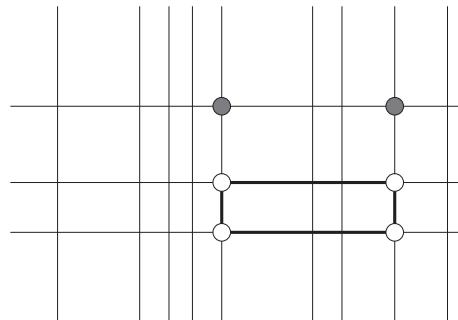
Biorąc $z = -1$ dostajemy żądaną związek

$$\sum_{k=0}^n k(-1)^k \binom{n}{k} = 0.$$

Na przykład dla czwartego rzędu trójkąta Pascala mamy

$$0 \cdot 1 - 1 \cdot 4 + 2 \cdot 6 - 3 \cdot 4 + 4 \cdot 1 = 0.$$

- Ćw. 2.11. Jest to zakamuflowany problem pokrycia odcinka o długości n kośćmi domina o długości 1 i 2.
- Ćw. 2.12. Pokolorowanie płaszczyzny dwoma kolorami oznacza, że każdy jej punkt ma jeden z dwóch kolorów. Rysujemy 3 dowolne linie poziome i 9 linii pionowych. Przecięcia każdej linii pionowej z poziomymi wyznaczają trójkę punktów. Różnie pokolorowanych trójkę punktów jest $2^3 = 8$, zatem na mocy zasady szufladkowej dwie z dziewięciu trójkę są pokolorowane jednakowo. Wybieramy te trójkę. W każdej z nich, znowu na mocy zasady szufladkowej, są po dwa jednakowo pokolorowane punkty. Wybieramy te punkty, co kończy konstrukcję (zob. rys. 7.6). Dla k kolorów wybraliśmy $k+1$ linii poziomych i $k^{k+1} + 1$ linii pionowych. Dalsze rozumowanie jest analogiczne do przypadku $k = 2$.
- Ćw. 2.13. Tworzymy n szufladek etykietowanych liczbą znajomych od 0 do $n-1$. Zauważmy najpierw, że niemożliwa jest sytuacja, w której szufladki 0 i $n-1$ są jednocześnie zajęte. Odpowiadałoby to bowiem sprzecznej sytuacji, gdzie są osoby, które nikogo nie znają, i osoby, które znają wszystkich. Mamy więc zapełnionych co najwyżej $n-1$ szufladek (o etykietach 1 do $n-1$ lub 0 do $n-2$). Ponieważ osób jest n , z zasady szufladkowej Dirichleta muszą być co najmniej dwie osoby o tej samej liczbie znajomych.



Rysunek 7.6: Na dowolnie pokolorowanej dwoma kolorami płaszczyźnie można, za pomocą zasady szufladkowej, wybrać prostokąt o wierzchołkach tego samego koloru

- Ćw. 2.14. Korzystamy z funkcji tworzącej dla współczynników dwumianowych. W pierwszym przypadku obliczamy $df(z)/dz$ dla $z = 1$, co daje

$$\sum_{k=0}^n k \binom{n}{k} = 2^{n-1} n.$$

W drugim przypadku obliczamy $\frac{d}{dz}[z \frac{d}{dz} f(z)]$ dla $z = 1$, co daje

$$\sum_{k=0}^n k^2 \binom{n}{k} = 2^{n-2} n(n+1).$$

- Ćw. 2.15. W przypadku pokrycia odcinka $a_n = a_{n-1} + a_{n-3}$, a analog wzoru (2.30) ma postać

$$a_n = \sum_{k=0}^{\lfloor n/3 \rfloor} \binom{n-2k}{k}.$$

Dla pokrycia okręgu $b_n = a_{n-1} + 2a_{n-3}$.

- Ćw. 2.16. Wzór rekurencyjny ma postać $a_n = 2a_{n-1} + 2a_{n-2}$, z warunkami początkowymi $a_1 = 2$, $a_2 = 6$. Rozwiążanie wynosi

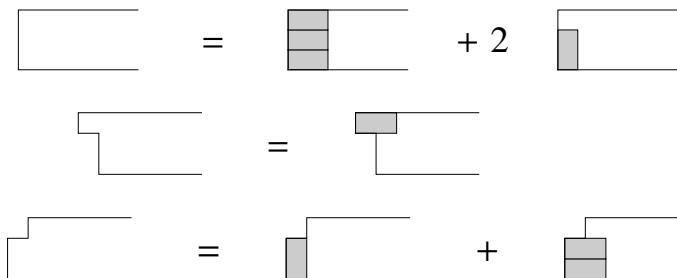
$$a_n = \frac{\sqrt{3} (1 - \sqrt{3})^n + (3 + 2\sqrt{3}) (1 + \sqrt{3})^n}{3 (1 + \sqrt{3})}.$$

- Ćw. 2.17. Idea rozwiązania jest analogiczna do problemów pokryć dominem z tą różnicą, że mamy teraz więcej konfiguracji lewego końca pokrycia parkietu. Wprowadzając stosowne oznaczenia dla konfiguracji według rys. 7.7, otrzymujemy układ równań rekurencyjnych

$$f_n = 2s_n + f_{n-2},$$

$$s_n = p_{n-1},$$

$$p_n = f_{n-1} + s_{n-1}.$$



Rysunek 7.7: Rekurencja dla ułożen parkietu z ćw. 2.17. W prawej części na górnym rysunku czynnik dwa pochodzi z możliwości ułożenia klepki na górze lub na dole

Z pierwszego równania mamy $s_n = (f_n - f_{n-2})/2$, a łącząc drugie i trzecie, otrzymujemy $s_n = f_{n-2} + s_{n-2}$, skąd

$$f_n = 4f_{n-2} - f_{n-4}. \quad (7.1)$$

Oczywiście, dla n nieparzystego $f_n = 0$, bo nie możemy pokryć podłogi o nieparzystej liczbie pól klepkami, które zawsze pokrywają ich parzystą liczbę. Warunki początkowe to $f_2 = 3$ i $s_2 = 1$, zatem z powyższych wzorów obliczamy $s_4 = 4$ i drugi warunek początkowy, $f_4 = 11$. Rozwiążanie rekurencji (7.1), którą zmieniamy na rekurencję rzędu 2 przez podstawienie $n = 2m$, ma postać

$$f_n = \frac{(3 + \sqrt{3})(2 - \sqrt{3})^{n/2} + (9 + 5\sqrt{3})(2 + \sqrt{3})^{n/2}}{6(2 + \sqrt{3})}, \quad (n = 2m).$$

Początkowe (parzyste) wyrazy, począwszy od $n = 2$, wynoszą

$$3, 11, 41, 153, 571, 2131, 7953, 29681, \dots$$

Inna metoda rozwiązywania problemów układania parkietu (za pomocą funkcji tworzących) omówiona jest w [3].

7.3 Rozdział 3

- Ćw. 3.1. W każdym rzucie dwiema kostkami prawdopodobieństwo wypadnięcia $\square\square$ wynosi $1/36$, a prawdopodobieństwo dowolnej innej konfiguracji $35/36$. Mamy $n = 24$ rzutów. Najpierw wybieramy, w których k rzutach wypadną $\square\square$, co możemy zrobić na C_k^n sposobów. Sytuacja wygrywająca to przynajmniej jedno wyrzucenie $\square\square$, tj. $k = 1, 2, \dots, n$, zatem

$$P = \sum_{k=1}^n \binom{n}{k} \left(\frac{1}{36}\right)^k \left(\frac{35}{36}\right)^{n-k} = 1 - \left(\frac{35}{36}\right)^n.$$

Zauważmy, że $\left(\frac{35}{36}\right)^n$ to prawdopodobieństwo zdarzenia dopełniającego, że nie wypadła ani raz para $\square\square$. Dla $n = 24$ otrzymujemy $P = 0.4914$, nieco mniej niż $1/2$. Prowadzący kasyno zarabia, ale różnica jest na tyle mała, że trudno to zauważyc przy krótkiej grze. Kawaler de Méré musiał być namiętnym hazardzistą, skoro spostrzegł empirycznie, że przegrywa. Zauważmy jeszcze, że już dla $n = 25$ grający ma większą szansę wygranej niż przegranej, $P_{25} = 0.5055$.

Gombaud nie rozumiał, dlaczego przegrywa w grę 24 rzutów, ponieważ w następującej grze 4 rzutów wygrywał: rzucamy kością 4 razy, jeśli choć raz wypadnie \square , wygrywamy. Rozumował następująco: w pojedynczym rzucie dwiema kościemi prawdopodobieństwo wypadnięcia $\square\square$ jest 6 razy mniejsze niż prawdopodobieństwo wypadnięcia \square w pojedynczym rzucie, co jest

słuszne. Jeśli więc skompensować ten czynnik 6 poprzez zwiększenie liczby rzutów z 4 do 24, gra 24 też powinna prowadzić do wygranej.

Obliczmy więc jak powyżej: w grze 4 kości prawdopodobieństwo wygranej wynosi $1 - (5/6)^4 = 0.5177 > 1/2$. Widać, że kawaler de Méré nie miał racji i powyższe „kompensowanie” nie działa! Paradoks polega na milczącym i fałszywym założeniu, że prawdopodobieństwo uzyskania wyniku w n rzutach jest proporcjonalne do n razy prawdopodobieństwo uzyskania wyniku w pojedynczym rzucie.

- Ćw. 3.2. Zastosujemy jawne zliczanie. Dla każdej pary graczy tworzymy tabelkę wszystkich możliwości, zaznaczając w polach, kto wygrywa. Na przykład dla graczy I i III mamy

	4	4	4	4	0	0
6	III	III	III	III	III	III
6	III	III	III	III	III	III
2	I	I	I	I	III	III
2	I	I	I	I	III	III
2	I	I	I	I	III	III
2	I	I	I	I	III	III

co daje prawdopodobieństwo zwycięstwa gracza III nad graczem I wynoszące $P(III > I) = 5/9$. Podobnie, $P(I > II) = 2/3$, $P(II > III) = 2/3$, $P(III > IV) = 2/3$, wreszcie $P(IV > I) = 2/3$. Pierwszy gracz (statystycznie) wygrywa z drugim, drugi z trzecim, trzeci z czwartym, a czwarty z pierwszym! Jest to ciekawy przykład relacji nieprzechodniej.

- Ćw. 3.3. Musimy jeszcze najpierw obliczyć $P(I > III) = 4/9$ oraz $P(II > IV) = 1/2$. Prawdopodobieństwo, że I wygra z losowo wybranym przeciwnikiem wynosi

$$P_I = 1/3(P(I > II) + P(I > III) + P(I > IV)) = \frac{1}{3} \left(\frac{1}{3} + \frac{4}{9} + \frac{2}{3} \right) = 13/27.$$

Podobnie, $P_{II} = 1/2$, $P_{III} = 14/27$ oraz $P_{IV} = 1/2$. Najlepszą kośćią jest więc kość III.

- Ćw. 3.4.

Dla talii o $n = 4k$ kartach mamy $|\Omega| = \binom{n}{5}$ wszystkich możliwości wyboru pięciu kart. Każda karta ma rangę (A, K, D, W, 10, ... itd.) oraz jeden z czterech kolorów. Poker to konfiguracja w jednym kolorze o kolejnych rangach. Dla pełnej talii mamy następujące możliwe pokery: A-K-D-W-10, K-D-W-10-9, ..., 6-5-4-3-2, 5-4-3-2-A (as liczy się też za 1), czyli łącznie 10 możliwości. Dla $k < 13$ nie ma sekwencji z asem na końcu, więc mamy

$k - 4$ możliwości, co ogólnie możemy zapisać, jako $k - 3 - m$, $m = 0$ dla $k = 13$, $m = 1$ dla $k < 13$. Poker może być w dowolnym z czterech kolorów, więc liczba wszystkich pokerów to $N_{\text{poker}} = 4(k - 3 - m)$. Kareta to cztery karty o tej samej randze. Wybieramy tę rangę na k sposobów, a pozostałą piątą kartę na $n - 4$ sposobów, co daje $N_{\text{kareta}} = k(n - 4)$. Full to trzy karty o tej samej randze plus dwie karty o tej samej randze. Rangę trójki można wybrać na k sposobów, kolory kart w trójce na C_3^4 sposoby, rangę pary na $(k - 1)$ sposobów (jedna jest już „zajęta” przez trójkę), a kolory kart w parze na C_2^4 sposoby. Łącznie mamy $N_{\text{full}} = k \binom{4}{3} (k - 1) \binom{4}{2}$. Kolor to układ pięciu kart w tym samym kolorze, który nie jest pokerem. Wybieramy więc jeden z czterech kolorów i pięć kart spośród k kart w tym kolorze, co daje $N_{\text{kolor}} = 4 \binom{k}{5} - N_{\text{poker}}$. Street to sekwencja kart o kolejnych rangach, która nie jest pokerem. Mamy więc $N_{\text{street}} = (k - m - 3)4^5 - N_{\text{poker}}$, gdzie czynnik $(k - m - 3)$ jest uzyskany podobnie, jak dla pokera, a 4^5 to możliwości wyboru koloru dla każdej z pięciu kart. Dla trójki mamy k możliwości wyboru rangi kart trójki, C_3^4 możliwości wyboru koloru dla trzech kart, następnie wybieramy na C_2^{k-1} możliwości dwie rangi spośród $(k - 1)$ dla pozostałych dwóch kart oraz ich kolory na 4^2 sposobów. Łącznie mamy więc $N_{\text{trójka}} = k \binom{4}{3} \binom{k-1}{2} 4^2$ możliwości. Dla dwóch par wybieramy dwie rangi dla par na C_2^k sposobów, kolory kart w parach na $(C_2^4)^2$ sposobów, rangę piątej karty na $(k - 2)$ sposoby, a jej kolor na 4 sposoby, co daje $N_{2 \text{ pary}} = \binom{k}{2} \binom{4}{2} (k - 2)4$. Wreszcie dla jednej pary wybieramy rangę na k sposobów, dwa spośród czterech możliwych kolorów kart w parze na C_2^4 sposoby, rangi pozostałych trzech kart na $\binom{k-1}{3}$ sposoby oraz ich

Tabela 7.1: Prawdopodobieństwa wylosowania układów w pokerze w grze taliami od dziewiątek ($k = 6$), siódemek ($k = 8$) i dwójków ($k = 13$)

	$k = 6, m = 1$	$k = 8, m = 1$	$k = 13, m = 0$
poker	0.000188	0.000078	0.000015
kareta	0.002823	0.001112	0.000240
full	0.016940	0.006674	0.001441
kolor	0.000376	0.001033	0.001965
street	0.047996	0.020261	0.003925
trójka	0.090344	0.053393	0.021129
dwie pary	0.203275	0.120133	0.047539
para	0.542067	0.533927	0.422569
pozostałe	0.095991	0.263388	0.501177

kolory na 4^3 sposobów. Łącznie daje to $N_{\text{para}} = \binom{k}{1} \binom{4}{2} \binom{k-1}{3} 4^3$.

Prawdopodobieństwa obliczamy jako $N_i/|\Omega|$. Wyniki liczbowe dla trzech typów talii, od dwójek, siódemek i dziewiątek, przedstawione są w tabeli 7.1. Zauważmy, że zmiana liczby kart w talii zmienia kolejność poszczególnych układów względem ich prawdopodobieństwa, np. dla $k=8$ kareta i full są bardziej prawdopodobne od koloru.

- Ćw. 3.5. Równa klasa graczy oznacza, że w każdej partii prawdopodobieństwo wygrania przez każdego gracza wynosi $1/2$. Założymy najpierw, że mamy dostatecznie długą grę (nikt nie wygrywa pod rząd), aby zauważać pewne prawidłowości. Przy stoliku siadają A i B . Jeśli wygrywa A , to wchodzi z ławki C . Teraz musi wygrać C , aby gra trwała dalej, więc A siada na ławce, a do gry przystępuje B . Teraz z kolei musi wygrać B , a do gry wchodzi ponownie A itd. Mamy więc sekwencję kolejnych zwycięzców $ACBACBACB\dots$ Podobnie, jeśli pierwszą partię wygrał B , to mamy ciąg zwycięzców $BCABCABCA\dots$ Rozważmy teraz prawdopodobieństwo, że wygra A . Sekwencje zwycięzców muszą się teraz kończyć przez $\dots AA$. Jeśli pierwszą partię wygrał A , to mamy możliwości

$$AA + ACBAA + ACBACBAA + ACB\dots ACBAA = \frac{1}{1-ACB}AA,$$

które formalnie zsumowaliśmy jako szereg geometryczny. Podobnie, jeśli pierwszy wygrał B , to mamy

$$BCAA + BCABCAA + BCA\dots BCAA = BCA \frac{1}{1-BCA}A.$$

Teraz wystarczy podstawić prawdopodobieństwa wygrania partii $A = B = C = 1/2$ i zsumować dwa powyższe wzory, co daje

$$P_A = \frac{1}{1-\frac{1}{8}\frac{1}{4}} + \frac{1}{8} \frac{1}{1-\frac{1}{8}\frac{1}{2}} \frac{1}{2} = \frac{5}{14}.$$

Z symetrii zagadnienia $P_B = P_A = \frac{5}{14}$. Dla gracza C otrzymujemy następujące sekwencje:

$$ACC + ACBACC + ACBA\dots CBACC = A \frac{1}{1-CBA}CC$$

oraz

$$BCC + BCABCC + BCAB\dots CABCC = B \frac{1}{1-CAB}CC.$$

Po podstawieniu prawdopodobieństw i zsumowaniu otrzymujemy $P_C = \frac{4}{14}$. Oczywiście, $P_A + P_B + P_C = 1$. Rozwiążanie pokazuje, że zaczynający turniej ma nieco większą szansę wygrania (w stosunku 5:4 do gracza początkowo pauzującego).

- Ćw. 3.6. Ze wzoru (3.26)

$$\bar{k} = \sum_{k=0}^n k P_k^n = \sum_{k=0}^n k \frac{1}{k!} \sum_{l=0}^{n-k} \frac{(-1)^l}{l!} = \sum_{k=0}^n \sum_{l=0}^{n-k} k \frac{1}{(k+l)!} \binom{k+l}{k} (-1)^l.$$

Wprowadźmy $l' = n - k - l$,

$$\bar{k} = \sum_{k=0}^n \sum_{l'=0}^{n-k} k \frac{1}{(n-l')!} \binom{n-l'}{k} (-1)^{n-k-l'},$$

a następnie zmieńmy kolejność sumowania. Granice sumowania odczytujemy z rys. 7.8. Otrzymujemy

$$\bar{k} = \sum_{l'=0}^n \frac{1}{(n-l')!} (-1)^{n-l'} \sum_{k=0}^{n-l'} k \binom{n-l'}{k} (-1)^k.$$

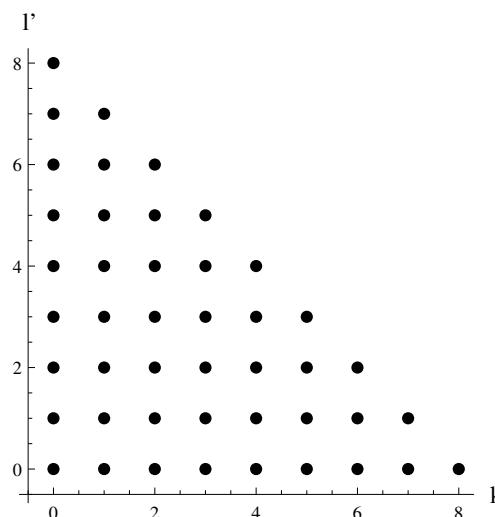
Na podstawie ćw. 2.10 suma po k w powyższym wzorze wynosi 0 dla $n-l' > 1$, zatem jedyny niezerowy przyczynek otrzymujemy dla $l' = n - 1$, $k = 1$, który daje

$$\bar{k} = \frac{1}{1!} (-1)^1 \binom{1}{1} (-1)^1 = 1.$$

Średnio, jeden kibic odzyskuje swój własny szalik.

- Ćw. 3.7. Oznaczmy szukane liczby usadowień n par jako M'_n . Podobnie jak w podrozdziale 3.5, zasada włączeń i wyłączeń prowadzi do wzoru

$$M'_n = \sum_{k=0}^n (-1)^k \binom{n}{k} |A'_k|, \quad (7.2)$$



Rysunek 7.8: Granice sumowania w rozwiązyaniu ćw. 3.6 dla $n = 8$: dla ustalonego l' sumujemy po k od 0 do $n - l'$, tj. otrzymujemy sumę $\sum_{l'=0}^n \sum_{k=0}^{n-l'} \dots$

Tabela 7.2: Liczba usadowień n małżeństw na przyjęciu, jeśli małżeństwa muszą siedzieć osobno

n	M'_n	$M'_n/(2n)!$
1	0	0
2	8	1/3
3	192	4/15
4	11904	0.295
5	1125120	0.310
6	153262080	0.320

gdzie $|A'_k|$ oznacza liczbę usadowień, w których k wybranych par siedzi razem (a inne mogą siedzieć razem lub nie). Anatomia $|A'_k|$ jest następująca:

- Wybór (podwójnych) miejsc dla wybranych k par – $c_k^{2n} = \frac{2n}{2n-k} \binom{2n-k}{k}$. Pamiętamy, że jest to liczba pokryć $2n$ punktów na okręgu za pomocą k kości domina i $2n - k$ kwadratów.
- Rozmieszczenie k par w k miejscach – $k!$.
- W każdej parze możliwe są dwa usadowienia, co daje – 2^k .
- Usadowienie pozostałych $2n - 2k$ osób – $(2n - 2k)!$.

Mamy więc

$$|A'_k| = c_k^{2n} k! 2^k (2n - 2k)! = 2n(2n - k - 1)! 2^k$$

oraz ostatecznie szukany wynik w postaci sumy:

$$M'_n = \sum_{k=0}^n (-1)^k \binom{n}{k} 2n(2n - k - 1)! 2^k. \quad (7.3)$$

Kilka pierwszych wartości dla liczb M'_n i dla prawdopodobieństwa uzyskania losowo właściwego usadowienia, $M'_n/|\Omega| = M'_n/(2n)!$, przedstawionych jest w tabeli 7.2.

- Ćw. 3.8. Niech $P(Z) = 0.12$ i $P(N) = 0.88$ oznaczają prawdopodobieństwa a priori pozostałe taksówki laptopa w taksówce, odpowiednio, zielonej lub niebieskiej, $P(\text{portier } Z|Z) = 0.8$ i $P(\text{portier } Z|N) = 0.2$ prawdopodobieństwa określenia przez portiera taksówki jako zielona pod warunkiem, że była ona, odpowiednio, zielona lub niebieska, a $P(Z|\text{portier } Z)$ i $P(N|\text{portier } Z)$ prawdopodobieństwa a posteriori, że taksówka była zielona lub niebieska pod warunkiem, że portier określił ją jako zieloną. Stosując wzór (3.34) otrzymujemy stosunek

$$\frac{P(N|\text{portier } Z)}{P(Z|\text{portier } Z)} = \frac{P(\text{portier } Z|N)}{P(\text{portier } Z|Z)} \frac{P(N)}{P(Z)} = \frac{0.2}{0.8} \frac{0.88}{0.12} = \frac{11}{6}.$$

Ponieważ $P(N|portier Z) + P(Z|portier Z) = 1$, otrzymujemy

$$P(N|portier Z) = \frac{11}{11+6} = \frac{11}{17} \simeq 0.647.$$

Tak więc, wbrew oświadczeniu portiera, jest większa szansa, że laptop został w niebieskiej taksówce i to mimo faktu, że portier myli kolory „tylko” w 20% przypadków.

- Ćw. 3.9. Prawdopodobieństwo a priori wskazania przez generała dowolnego pudła wynosi $\frac{1}{3}$. Niech pudło, które wskazał generał ma etykietę 1, a pudło otwarte przez prowadzącego etykietę 2. Jeśli „Beryl” jest w pudle 1, to prowadzący może otworzyć (z jednakowym prawdopodobieństwem) pudła 2 lub 3, a więc $P(prow\ 2|1) = \frac{1}{2}$. Jeśli jednak karabin jest w pudle 3, to prowadzący może otworzyć jedynie pudło 2 i wtedy $P(prow\ 2|3) = 1$. Ze wzoru (3.34)

$$\frac{P(3|prow\ 2)}{P(1|prow\ 2)} = \frac{P(prow\ 2|3)P(3)}{P(prow\ 2|1)P(1)} = \frac{1}{1/2} \frac{1/3}{1/3} = 2,$$

skąd $P(3|prow\ 2) = \frac{2}{3}$ i $P(1|prow\ 2) = \frac{1}{3}$. Generał powinien więc zmienić wskazanie z 1 na 3, bo w ten sposób zdubluje swoja szansę. Nieco paradoksalnie wyglądający rezultat wynika z faktu, że prowadzący, otwierając puste pudło, udziela generałowi cennej informacji!

- Ćw. 3.10. (a) Oznaczając prawdopodobieństwo, że mysz wyjdzie na wolność, będąc początkowo w pomieszczeniu i jako $P(i)$ i korzystając, podobnie jak w zadaniu o ruinie gracza, ze wzoru na prawdopodobieństwo całkowite, możemy zapisać

$$\begin{aligned} P(0) &= 0, \\ P(1) &= \frac{1}{3}P(0) + \frac{1}{3}P(2) + \frac{1}{3}P(4), \\ P(2) &= \frac{1}{2}P(1) + \frac{1}{2}P(3), \\ P(3) &= \frac{1}{3}P(2) + \frac{1}{3}P(4) + \frac{1}{3}P(5), \\ P(4) &= \frac{1}{2}P(1) + \frac{1}{2}P(3), \\ P(5) &= 1. \end{aligned}$$

Ten układ równań liniowych można bez problemu rozwiązać „na piechotę”, jest jednak prostsza metoda. Z symetrii zagadnienia (zob. rys 3.10) wynika, że $P(2) = P(4) = 1/2$. Stąd drugie i czwarte z powyższych równań dają $P(1) = 1/3$ i $P(3) = 2/3$.

(b) Ze wzoru Bayesa, prawdopodobieństwo a posteriori, że mysz startowała z pomieszczenia $i = 1, 2, 3, 4$, wynosi $P(i)/2$. Jest tak, ponieważ prawdopodobieństwo a priori włożenia przez laboranta myszy do pomieszczenia 1, 2, 3, 4, wynosi $1/4$, a prawdopodobieństwo a priori ucieczki wynosi $1/2$.

7.4 Rozdział 4

- Ćw. 4.1. Pierwsze 3 wzory (4.2-4.5) są trywialne. Ostatni wynika z faktu, że dwa elementy łączymy w cykl (lub podzbiór), co możemy zrobić na C_2^n sposobów.
- Ćw. 4.2. Postępujemy indukcyjnie względem n , stosując wzory z Tw. 4.1. Mamy

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

oraz

$$\begin{aligned} \sum_{k=0}^{n+1} \begin{bmatrix} n+1 \\ k \end{bmatrix} &= \begin{bmatrix} n+1 \\ 0 \end{bmatrix} + \sum_{k=0}^{n-1} \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} + \begin{bmatrix} n+1 \\ n+1 \end{bmatrix} \\ &= 0 + \sum_{k=0}^{n-1} \begin{bmatrix} n \\ k \end{bmatrix} + \sum_{k=0}^{n-1} n \begin{bmatrix} n \\ k+1 \end{bmatrix} + 1 = (n+1) \sum_{k=0}^n \begin{bmatrix} n \\ k \end{bmatrix}. \end{aligned}$$

Jest to ta sama rekurencja, co dla silni, skąd $\sum_{k=0}^n \begin{bmatrix} n \\ k \end{bmatrix} = n!$. Wynik zgadza się z faktem, że wszystkie $n!$ permutacji n -elementowych można jednoznacznie rozłożyć na cykle.

- Ćw. 4.3. (a) Jest to zakamuflowany problem podziału zbioru na podzbiory, a zatem liczba konfiguracji dana jest liczbą Bella B_n .
 - (b) Dla ustalenia uwagi weźmy $n = 5$ i ponumerujmy matrioszki od najmniejszej, 1, do największej, 5. Jedna konfiguracja, którą oznaczymy jako (5), zawiera wszystkie lalki jedna w drugiej, czyli widzimy tylko lalkę 5. Następnie mamy $\binom{4}{1} = 4$ konfiguracje, gdzie widzimy dwie lalki: (54), (53), (52), (51). Kolejno, są $\binom{4}{2} = 6$ konfiguracje, gdzie widzimy 3 lalki: (543), (542), (541), (532), (531), (521), następnie $\binom{4}{3} = 4$ konfiguracje, gdzie widzimy 4 lalki: (5432), (5431), (5421), (5321), wreszcie jedna konfiguracja, gdzie widzimy wszystkie 5 lalek: 54321. Ogólnie mamy zatem

$$\sum_{k=0}^{n-1} \binom{n-1}{k} = 2^{n-1}$$

konfiguracji. Nota bene dokładnie tyle samo konfiguracji mamy w przypadku, gdy dopuścimy „nielegalne” konfiguracje z rys. 4.14, ponieważ nie ma znaczenia, czy w środku lalki są włożone jedna w drugą, czy też nie, albowiem tego nie widzimy!

- Ćw. 4.4. Gdyby wiaderka były nieroróżnialne, mielibyśmy $\binom{n}{k}$. Rozróżnialność wiaderek powoduje, że możemy je dodatkowo ustawić na $k!$ sposobów, co łącznie daje

$$k! \left\{ \begin{array}{c} n \\ k \end{array} \right\}.$$

- Ćw. 4.5. Najpierw musimy włożyć po jednej piłce do każdego wiaderka. Pozostałe $n - k$ piłek możemy włożyć (patrz zad. 2.7) na

$$\left(\begin{array}{c} (n-k)+k-1 \\ n-k \end{array} \right) = \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right)$$

sposobów.

- Ćw. 4.7. Rozkłady liczby $n + k$ na dokładnie k składników mają w górnym wierszu diagramów Ferrersa rząd o długości k , a w niższe rzędy z konstrukcji mają długość co najwyżej k . Usuwając najwyższy rząd, dostajemy zatem diagramy rozkładów liczby n na co najwyżej k składników.
- Ćw. 4.8. Funkcja generująca dla rozkładu, w którym liczby parzyste się nie powtarzają (a składniki nieparzyste mogą się powtarzać), wynosi

$$\begin{aligned} f(z) &= (1+z+z^2+z^3+\dots)(1+z^2)(1+z^3+z^6+z^9+\dots)(1+z^4)\dots \\ &= \frac{1+z^2}{1-z} \frac{1+z^4}{1-z^3} \frac{1+z^6}{1-z^5} \frac{1+z^8}{1-z^7} \dots \\ &= \frac{(1+z)(1+z^2)}{(1+z)(1-z)} \frac{(1+z^2)(1+z^4)}{(1+z^2)(1-z^3)} \frac{(1+z^3)(1+z^6)}{(1+z^3)(1-z^5)} \frac{(1+z^4)(1+z^8)}{(1+z^4)(1-z^7)} \dots \\ &= \frac{(1+z+z^2+z^3)}{(1-z^2)} \frac{(1+z^2+z^4+z^6)}{(1+z^2)(1-z^3)} \frac{(1+z^3+z^6+z^8)}{(1+z^3)(1-z^5)} \dots \\ &= (1+z+z^2+z^3)(1+z^2+z^4+z^6)(1+z^3+z^6+z^8)\dots, \end{aligned}$$

gdzie mianownik „zwinał się” podobnie, jak w wyprowadzeniu wzoru (4.25). Ostatnia linijka zawiera funkcję tworzącą dla partycji, gdzie żaden składnik nie powtarza się więcej niż 3 razy.

- Ćw. 4.9. Oznaczmy partycje niezawierające jedynki jako r_n . Funkcja rozkładu dla partycji dla r_n wynosi

$$\begin{aligned} \sum_{k=0}^{\infty} r_k x^k &= \frac{1}{1-x^2} \frac{1}{1-x^3} \frac{1}{1-x^4} \dots = (1-x) \frac{1}{1-x} \frac{1}{1-x^2} \frac{1}{1-x^3} \frac{1}{1-x^4} \dots \\ &= \sum_{k=0}^{\infty} p_k x^k - \sum_{k=0}^{\infty} r_k x^{k+1} = \sum_{k=0}^{\infty} p_k x^k - \sum_{k=1}^{\infty} r_{k-1} x^k, \end{aligned}$$

skąd dla $n > 0$

$$r_n = p_n - p_{n-1}.$$

- Ćw. 4.10. Pierwsze z równań (4.21) wynika w bezpośredni sposób z rekurencji (4.20) dla $k = 2$:

$$p(n, 2) = p(n-1, 1) + p(n-2, 2) = 1 + p(n-2, 2),$$

z warunkami początkowymi $p(1, 2) = 0$, $p(2, 2) = 1$. Równanie stanoi niezależne rekurencje dla wyrazów parzystych, $n = 2m$, o rozwiązaniu $p(2m, 2) = m$ (suma m jedynek) i dla wyrazów nieparzystych $n = 2m + 1$, o rozwiązaniu $p(2m+1, 2) = m$. Ponieważ

$$m = \left\lfloor \frac{2m}{2} \right\rfloor = \left\lfloor \frac{2m+1}{2} \right\rfloor,$$

w obu przypadkach $m = \left\lfloor \frac{n}{2} \right\rfloor$.

Drugie z równań (4.21) wykazujemy w analogiczny sposób. Mamy teraz rekurencję

$$p(n, 3) = p(n-1, 2) + p(n-3, 3) = \left\lfloor \frac{n-1}{2} \right\rfloor + p(n-3, 3),$$

która stanowi 3 niezależne od siebie rekurencje dla $n = 1, 4, 7, 10 \dots$, $n = 2, 5, 8, 11 \dots$ i $n = 3, 6, 9, 12 \dots$

- Ćw. 4.12.

Zadanie to jest przykładem zagadnienia znanego jako *problem rozmieniania*, omówionego dokładnie w [3]. Jest to problem partycji, gdzie zbiór składników, na które rozkładamy liczbę, jest pewnym skończonym podzbiorem zbioru liczb naturalnych. Rozumowanie analogiczne do przedstawionego w podrozdziale 4.5 prowadzi do funkcji tworzącej

$$\begin{aligned} g(z) &= (1+z+z^2+z^3+\dots)(1+z^2+z^4+\dots)(1+z^5+z^{10}+\dots) \\ &= \frac{1}{(1-z)(1-z^2)(1-z^5)}. \end{aligned}$$

Pragniemy teraz znaleźć współczynniki rozwinięcia $g(z) = \sum_n a_n z^n$, co da odpowiedź na postawione pytanie. Oczywiście, możemy z leistwa użyć komputera celem wymnożenia wielomianów i odczytać wynik ze współczynnikami przy z^n , ale bądźmy nieco ambitniejsi. Poniższa metoda przedstawiona szczegółowo jest w [3]. Korzystając z tożsamości

$$\begin{aligned} 1-z^5 &= (1-z)(1+z+z^2+z^3+z^4), \\ 1-z^{10} &= (1-z^2)(1+z^2+z^4+z^6+z^8), \\ (1-z^5)(1+z^5) &= 1-z^{10}, \end{aligned}$$

możemy zapisać

$$\begin{aligned} g(z) &= \frac{(1+z+z^2+z^3+z^4)}{(1-z^5)(1-z^2)(1-z^5)} \\ &= \frac{(1+z+z^2+z^3+z^4)(1+z^2+z^4+z^6+z^8)}{(1-z^5)^2(1-z^{10})} \\ &= \frac{(1+z+z^2+z^3+z^4)(1+z^2+z^4+z^6+z^8)(1+z^5)^2}{(1-z^{10})^3}. \end{aligned}$$

Licznik pieczołowicie rozwijamy, dostając

$$\begin{aligned} L(z) = \sum_m l_m z^n &= 1 + z + 2z^2 + 2z^3 + 3z^4 + 4z^5 + 5z^6 + 6z^7 + 7z^8 + 8z^9 \\ &\quad + 7z^{10} + 8z^{11} + 7z^{12} + 8z^{13} + 7z^{14} + 6z^{15} + 5z^{16} + 4z^{17} + 3z^{18} + 2z^{19} \\ &\quad + 2z^{20} + z^{21} + z^{22}. \end{aligned} \tag{7.4}$$

Następnie korzystając ze wzoru dwumianowego, możemy zapisać

$$(1-z^{10})^{-3} = \sum_{k=0}^{\infty} \binom{-3}{k} (-z^{10})^k = \sum_{k=0}^{\infty} \binom{k+2}{2} z^{10k},$$

gdzie skorzystaliśmy z tożsamości dla współczynnika dwumianowego o ujemnym górnym indeksie,

$$\binom{-n}{k} = (-1)^k \binom{k-n-1}{-n-1}.$$

Mamy więc

$$f(z) = \sum_n a_n z^n = L(z) \sum_{k=0}^{\infty} \binom{k+2}{2} z^{10k}.$$

Zapisując n w postaci $n = 10i + j$, gdzie $0 \leq j \leq 9$, otrzymujemy równość z sumą po m i k , ale z ograniczeniem $10i + j = 10k + m$ wynikającym z przyrównania potęg z , mianowicie:

$$a_{10i+j} = \sum_{m,k} l_m \binom{k+2}{2} \delta_{10i+j, 10k+m}.$$

Mamy przypadki $m = j$, $k = i$, następnie $m = j + 10$, $k = i - 1$ oraz $m = j + 20$, $k = i - 2$. Większych wartości m nie ma, bo maksymalna potęga z w $L(z)$ to 22. Możemy teraz napisać ostateczny wynik

$$a_{10i+j} = l_j \binom{i+2}{2} + l_{j+10} \binom{i+1}{2} + l_{j+20} \binom{i}{2}.$$

Z (7.4) wyczytujemy stosowne współczynniki l_j , l_{j+10} , l_{j+20} , przemnażamy je przez odpowiednie symbole Newtona i dostajemy pożądaną liczbę. Na przykład

$$a_{66} = l_6 \binom{8}{2} + l_{16} \binom{7}{2} + l_{26} \binom{6}{2} = 5 \frac{8 \cdot 7}{2} + 5 \frac{7 \cdot 6}{2} + 0 = 245,$$

czyli kioskarz ma aż 245 sposobów na wydanie 66 złotych reszty monetami 1, 2 i 5 złotowymi.

- Ćw. 4.13. W rozwiążaniu tego wariantu problemu nie musimy męczyć się w wymnażaniem funkcji tworzących! Największa kwota, jaką kioskarz może wydać, to $9 \cdot 5 + 9 \cdot 2 + 9 \cdot 1 = 72$ zł, co oczywiście da się zrobić tylko na jeden sposób. Kwota 66 zł jest o 6 zł mniejsza, zatem wydając kioskarz może odliczyć sobie te 6 zł, co może zrobić na następujących 5 sposobów: $5 + 1$, $2 + 2 + 2$, $2 + 2 + 1 + 1$, $2 + 1 + 1 + 1 + 1$, $1 + 1 + 1 + 1 + 1 + 1$. Tym samym na tyle samo sposobów może wydać 66 zł.
- Ćw. 4.14. (a) Oznaczmy liczbę konfiguracji posiadających n jednogroszówek w najniższym rzędzie jako c_n . Jeśli wyższy rząd ma $n - k$ monet, możemy go ułożyć (przesuwając) na k sposobów. Daje to rekurencję

$$c_n = c_{n-1} + 2c_{n-2} + 3c_{n-3} + \cdots + (n-1)c_1, \quad c_1 = 1.$$

Porównując z wynikiem ćw. 1.5 (d), dostajemy $c_n = F_{2n-1}$, tj. nieparzyste liczby Fibonacciego.

(b) Ten wariant zadania jest trudniejszy. Oznaczmy przez g_k^n konfiguracje n jednogroszówek posiadające w najniższym rzędzie k monet. Liczby te spełniają następującą rekurencję:

$$g_k^n = \sum_{l=a}^b (k-l) g_l^{n-k} \quad (\text{dla } n > k), \quad g_n^n = 1.$$

Postać ta wynika z faktu, że nad dolnym rzędem, który ma długość k , leży łącznie $n - k$ monet. Dolny rząd tej górnej konfiguracji, o długości l , ma co najwyżej $b = \min(k-1, n-k)$ monet, a co najmniej $a = \lceil \frac{-1+\sqrt{1+8(n-k)}}{2} \rceil$. Jest tak, ponieważ liczba trójkątna $l(l+1)/2$ musi być większa lub równa $(n-k)$, a rozwiązanie na l daje napisany warunek. Czynnik $(k-l)$ mnożący g_l^{n-k} wynika z liczby możliwości ułożień (przesunięć) górnej konfiguracji na dolnym rzędzie o długości k . Wykonując sumowanie po k ,

$$\sum_{k=\lceil(-1+\sqrt{1+8n})/2\rceil}^n g_k^n,$$

uzyskujemy całkowitą liczbę konfiguracji n jednogroszówek. Początkowe wartości ciągu wynoszą

$$1, 1, 2, 3, 5, 8, 12, 18, 26, 38, 53, 75, 103, 142, 192, 260, 346, 461, 607, 797, \dots$$

7.5 Rozdział 5

- Ćw. 5.2. Element (i, j) iloczynu dwóch macierzy A i B , wynosi $(AB)_{ij} = \sum_k A_{ik}B_{kj}$. Postępując rekurencyjnie, dostajemy

$$(S^m)_{ij} = \sum_{k_1}^n \sum_{k_2}^n \cdots \sum_{k_{m-1}}^n S_{ik_1} S_{k_1 k_2} \dots S_{k_{m-1} j},$$

gdzie n jest liczbą wierzchołków grafu. Z konstrukcji macierzy sąsiedztwa $S_{ab} = 1$, gdy wierzchołek a jest połączony z b , a 0 w przeciwnym razie. W związku z tym iloczyn $S_{ik_1} S_{k_1 k_2} \dots S_{k_{m-1} j} = 1$ wtedy i tylko wtedy, gdy $(i, k_1, k_2, \dots, k_{m-1}, j)$ jest drogą. Sumowanie po k_l w $(S^m)_{ij}$ daje więc liczbę wszystkich możliwych dróg o długości m z i do j .

Założenie, że graf jest prosty jest istotne, bo obecność pętli lub krawędzi wielokrotnych komplikuje interpretację.

- Ćw. 5.3. Korzystając z poprzedniego zadania dostajemy natychmiast, że element diagonalny $(S^2)_{ii}$ równy jest liczbie dróg o długości 2 o początku i końcu w wierzchołku i . Dla grafu nieskierowanego każda z tych dróg prowadzi do wierzchołka sąsiadniego z i , po czym wraca po tej samej krawędzi. W związku z tym $(S^2)_{ii}$ jest stopniem wierzchołka i . Suma tych stopni, na mocy lematu o uściskach dłoni 5.1, jest podwojoną liczbą krawędzi grafu k , tj. $\text{Tr } S^2 = 2 \cdot k$.

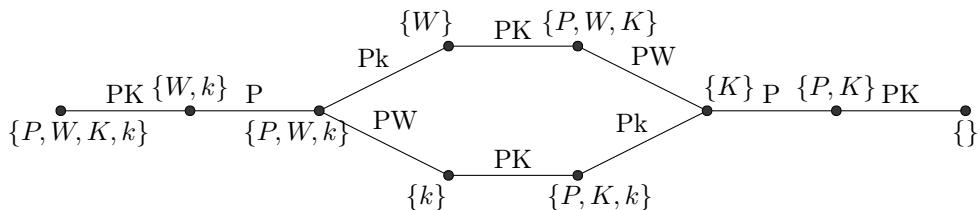
Podobnie, element S_{ii}^3 jest liczbą dróg o długości 3 o początku i końcu w wierzchołku i . Drogi te muszą być trójkątami. Ponieważ każdy trójkąt może być obchodzony w dwóch kierunkach oraz zawiera 3 wierzchołki (zliczając po wierzchołkach, policzylibyśmy trójkąty potrójnie), $\text{Tr } S^3 = 2 \cdot 3 \cdot N_3$, gdzie N_3 jest liczbą trójkątów w grafie.

- Ćw. 5.4 $\binom{n}{3} = n(n-1)(n-2)/6$.
- Ćw. 5.5. Wprowadźmy oznaczenia P-przewoźnik, W-wilk, K-koza, k-kapusta. Niech na początku przewoźnik wraz z dobytkiem znajduje się na lewym brzegu rzeki. Oznaczmy wierzchołki grafu (stany) przez zbiór obiektów znajdujących się na lewym brzegu rzeki. Stan początkowy to $\{P, W, K, k\}$, a końcowy $\{\}$. Stany pośrednie to stany po zakończeniu każdej przeprawy. Wszystkich stanów jest $2^4 = 16$. Wśród nich 10 jest dozwolonych, tzn. nikt nikogo nie zjada, oraz 6 wzbronionych, gdzie niepilnowany wilk zjada kozę lub niepilnowana koza kapustę. Stany dozwolone to

$$\begin{aligned} & \{P, W, K, k\}, \{P, W, K\}, \{P, W, k\}, \{P, K, k\}, \\ & \{P, K\}, \{W, k\}, \{W\}, \{K\}, \{k\}, \{\}, \end{aligned}$$

a wzbronione

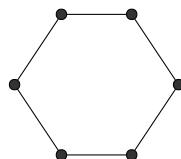
$$\{W, K, k\}, \{W, K\}, \{K, k\}, \{P\}, \{P, W\}, \{P, k\}.$$



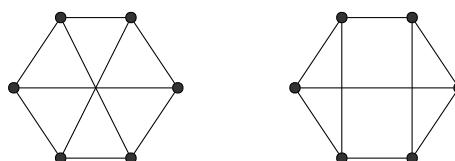
Rysunek 7.9: Graf wilka, kozy i kapusty

Przy powyższym podziale musimy rzec jasna zwracać uwagę na niewypisane obiekty znajdujące się na prawym brzegu, będące dopełnieniami podanych zbiorów. Na przykład $\{P, W\}$ oznacza, że na lewym brzegu znajduje się przewodnik z wilkiem, a na prawym koza zajada się kapustą! Ponieważ stany wzbronione nie są interesujące, bo z definicji nie mogą występować w rozwiązańiu, utwórzmy graf ze stanów dozwolonych (zob. rys. 7.9). Krawędzie etykietowane są parą przewożonych obiektów, wśród których oczywiście zawsze jest przewoźnik. Z grafu widzimy natychmiast, że są dwa optymalne rozwiązania wymagające siedmiu przepraw przez rzekę.

- Ćw. 5.7. Na mocy Tw. Diraca szukane grafy są hamiltonowskie. Możemy więc zacząć konstrukcję od grafu zawierającego cykl Hamiltona:



Następnie dorysowujemy w ten sześciokąt cięciwy w taki sposób, aby stopień każdego wierzchołka wyniósł 3. Rozważenie wszystkich możliwości nie nastręcza trudności i prowadzi do następujących dwóch nieizomorficznych grafów:



- Ćw. 5.8. Zauważmy, że każdy graf G ma atomy wodoru (wierzchołki stopnia jeden) na końcach krawędzi (liście), w związku z tym jeśli je usuniemy, to pozostały graf G' złożony jedynie z atomów węgla jest spójny. Innymi słowy, każdy atom C sąsiaduje z jakimś innym atomem C . W związku z tym rozważając grafy, możemy rysować tylko atomy C , rozumiejąc, że graf jest uzupełniony przez atomy H do grafu G w taki sposób, aby stopień każdego wierzchołka C wyniósł 4.

Pokażemy najpierw, że jeśli G jest drzewem, to przedstawia związek $C_k H_{2k+2}$. Niech cząsteczka węglowodoru zawiera k atomów węgla. Suma



Rysunek 7.10: Izometry C_6H_{14} . Wierzchołki oznaczają atomy węgla

wszystkich stopni wierzchołków C wynosi więc $4k$. Jeśli graf G jest drzewem, to G' jest również drzewem, zatem G' posiada $k - 1$ krawędzi. W związku z tym łącznie $2(k - 1)$ stopni wierzchołków C jest „zużytych” na wiązania $C - C$. Na wiązania $C - H$ pozostaje więc $4k - 2(k - 1) = 2k + 2$ stopni i właśnie tyle atomów H jest dołączonych, tworząc cząsteczkę C_kH_{2k+2} . Teraz udowodnimy, że jeśli G nie jest drzewem, to przedstawia cząsteczkę C_kH_n o $n < 2k + 2$. Pamiętamy, że wśród grafów spójnych o k wierzchołkach drzewo ma najmniejszą możliwą liczbę krawędzi. W związku z tym jeśli G' nie jest drzewem, to jest połączone $m > k - 1$ krawędziami, wobec czego może dołączyć $4k - 2m < 2k + 2$ atomów H . Rys. 7.10 przedstawia izometry węglowodoru C_6H_{14} (atomów wodoru nie rysujemy).

- Ćw. 5.10. Domki oraz {elektrownia, gazownia, ujęcie wody} wraz z połączeniami tworzą graf $K_{3,3}$, który nie jest planarny. Zatem nie jest możliwe ustalenie połączeń bez krzyżowania linii zasilania.
- Ćw. 5.9. Wszystkich grafów jest $2^{n(n-1)/2}$ (zob. podrozdział 5.6).
 - (a) Graf cykliczny C_n możemy wybrać na $n!/(2n)$ sposobów (pierwszy wierzchołek na n sposobów, drugi na $(n - 1)$ itd., ale musimy podzielić przez liczbę wierzchołków n oraz dwie orientacje). Prawdopodobieństwo wynosi więc $(n - 1)!/2^{1+n(n-1)/2}$. Dla kolejnych n , począwszy od 3, otrzymujemy szybko malejący ciąg $1/8, 3/64, 3/256, 15/8192, \dots$
 - (b) Wydzielimy arbitralnie ze zbioru n wierzchołków maksymalną liczbę rozłącznych podzbiorów o trzech wierzchołkach każdy. Podzbiorów tych jest $\lfloor n/3 \rfloor$. Prawdopodobieństwo wylosowania trójkąta w każdym z podzbiorów wynosi $(1/2)^3 = 1/8$. Ponieważ podzbiory są rozłączne, losowania trójkątów są niezależne. W związku z tym prawdopodobieństwo niewylosowania trójkąta w żadnym z podzbiorów wynosi $(7/8)^{\lfloor n/3 \rfloor}$, tym samym prawdopodobieństwo wylosowania grafu z trójkątem w dowolnym podzbiorze (zdarzenie przeciwe) wynosi $P = 1 - (7/8)^{\lfloor n/3 \rfloor}$. Prawdopodobieństwo wylosowania grafu zawierającego trójkąt bez ograniczenia wynikającego z powyższego podziału na podzbiory jest większe od P . Ponieważ $\lim_{n \rightarrow \infty} P = 1$, prawdopodobieństwo wylosowania grafu zawierającego trójkąt też dąży do 1.
- Ćw. 5.11. W języku grafów zadanie to można sformułować następująco: ile kolorów potrzeba do pokolorowania krawędziowego grafu $K_{n,m}$, $m \geq n$. Kolor oznacza tu dany taniec. Wystarczy m kolorów, a więc m tańców.

- Ćw. 5.12. Postępując analogicznie jak w wyprowadzeniu wzoru (5.9), ale z równaniem $w - k + s = 0$ (torus), otrzymujemy warunek

$$s \left(\frac{p}{m} - \frac{p}{2} + 1 \right) = 0.$$

Ponieważ $s > 0$, dostajemy stąd

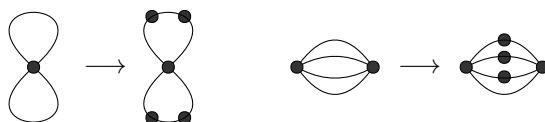
$$2p - pm + 2m = 4 - (p - 2)(m - 2) = 0.$$

Jedyne rozwiązania to $p = 6$, $m = 3$, następnie $p = 4$, $m = 4$ oraz $p = 3$, $m = 6$. Zauważmy, że s jest nieokreślone.

Interpretacja topologiczna powyższych rozwiązań jest pouczająca. Weźmy przypadek $p = 4$, $m = 4$, który odpowiada siatce o czworokątnych oczkach. Taką siatką możemy opleść torus. Pozostałe dwa przypadki odpowiadają siatce o oczkach sześciokątnych i trójkątnych.

- Ćw. 5.14. Mamy pokazać, że w n wymiarach hiper kostka ma $w_n = 2^n$ wierzchołków (co jest oczywiste) i $k_n = n2^{n-1}$ krawędzi. Postąpimy indukcyjnie. Dla $n = 1$ mamy 2 wierzchołki i jedną krawędź (zob. rys. 5.62), więc twierdzenie jest w tym przypadku spełnione. W $n+1$ wymiarach tworzymy wierzchołki Q_{n+1} przez dopisanie na końcu ciągu współrzędnych każdego wierzchołka Q_n elementu 0 lub 1. W ten sposób liczba wierzchołków się podwaja i mamy $w_{n+1} = 2w_n = 2^{n+1}$. Metoda indukcyjna polega więc de facto na „sklejaniu” dwóch hiper kostek Q_n w hiper kostkę Q_{n+1} . Konstrukcja pokazuje, że liczba krawędzi Q_{n+1} wynosi $k_{n+1} = k_n + k_n + 2^n$. Pierwszy składnik to liczba krawędzi podgrafu Q_{n+1} , gdzie ostatnia współrzędna każdego wierzchołka wynosi 0. Ten podgraf to jedna ze sklejanych hiper kostek Q_n , a więc z definicji posiada k_n krawędzi. Podobnie, drugi składnik to liczba krawędzi podgrafu o wierzchołkach, gdzie ostatnia współrzędna wynosi 1, czyli ponownie k_n . Trzeci składnik to liczba par sąsiednich wierzchołków postaci $(a_1, a_2, \dots, a_n, 0)$ i $(a_1, a_2, \dots, a_n, 1)$, których jest 2^n . Mamy więc $k_{n+1} = 2n2^{n-1} + 2^n = (n+1)2^n$, co kończy dowód pierwszej części zadania. Sumując współrzędne danego wierzchołka, otrzymujemy liczbę parzystą lub nieparzystą. Definiujemy *sygnaturę* wierzchołka jako +1, gdy suma współrzędnych jest parzysta, oraz -1, gdy jest nieparzysta. Z konstrukcji hiper kostki wynika, że wierzchołki sąsiednie mają przeciwnie sygnatury, a zatem wierzchołki o tych samych sygnaturach nie są sąsiednie. Oznacza to, że Q_n jest grafem dwudzielnym, gdzie rozłączne zbiory wierzchołków są określone sygnaturą.

- Ćw. 5.15. Rysujemy najpierw graf cykliczny C_5 w układzie pięciokąta. Następnie dobudowujemy przekątne, począwszy od jednej (jedna nieizomorficzna możliwość), dwóch (dwie możliwości), trzech (dwie możliwości), czterech (jedna możliwość) i pięciu (jedna możliwość). Łącznie mamy 8 nieizomorficznych grafów, zgodnie z tabelą 5.2.



Rysunek 7.11: Konstrukcja grafów z ćw. 5.16

- Ćw. 5.16. Stopień każdego wierzchołka musi być parzysty, a maksymalnie wynosi 4, w związku z czym wynosi 2 lub 4. Jeśli wszystkie wierzchołki mają stopień 2, to mamy graf cykliczny C_5 . Jeśli wszystkie mają stopień 4, to mamy graf pełny K_5 . Pozostają do rozważenia przypadki pośrednie. Wierzchołki o stopniu 2 możemy „odłożyć na bok” i rozważać najpierw grafy zbudowane z wierzchołków o stopniu 4.

Dla grafu o jednym wierzchołku stopnia 4 mamy sytuację z lewej części rys. 7.11. Następnie wierzchołki o stopniu 2 „nanizujemy” na krawędzie w taki sposób, aby nie pozostały pętle ani krawędzie wielokrotne. Pozostawia to jedną możliwość. Dla grafu o dwóch wierzchołkach stopnia 4 sytuacja jest ukazana w prawej części rys. 7.11. Dla grafu o trzech wierzchołkach stopnia 4 nie jest możliwe uzupełnienie krawędzi dwoma pozostałymi wierzchołkami stopnia 2 w taki sposób, by powstały graf był grafem prostym. Mamy więc 4 klasy nieizomorficznych prostych grafów eulerowskich o pięciu wierzchołkach, zgodnie z tabelą 5.2.

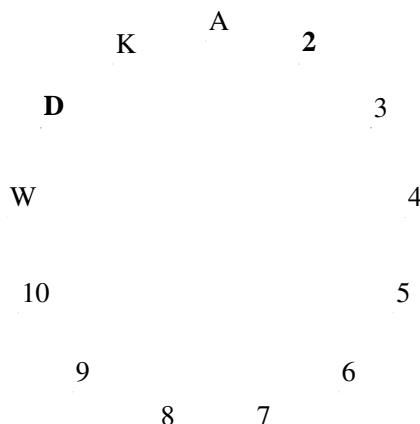
- Ćw. 5.17. Tworzymy graf z wagami krawędziowymi, w którym krawędziami są ulice, wagami ich długości, a wierzchołkami skrzyżowania. Jeśli graf jest eulerowski, to cykl Eulera jest rozwiązaniem problemu. Jeśli graf nie jest eulerowski, to na mocy Tw. 5.2 zawiera wierzchołki stopnia nieparzystego. Z lematu o uściskach dłoni sumaryczny stopień wszystkich wierzchołków jest parzysty, zatem liczba wierzchołków stopnia nieparzystego musi być parzysta. Łączymy parami te n wierzchołków dodatkowymi krawędziami prowadzącymi wzduż istniejących ulic. Spośród tych sposobów wybieramy ten o najkrótszej sumarycznej długości nowych krawędzi. Powstały graf jest eulerowski, gdyż ma wszystkie wierzchołki stopnia parzystego. Cykl Eulera tego grafu jest rozwiązaniem problemu chińskiego listonosza.
- Ćw. 5.18. Wynikiem tej operacji jest trójkąt Sierpińskiego! Twierdzenie to pochodzi od Lucasa.
- Ćw. 5.23. Wbrew pozorom zadanie jest trywialne. Wystarczy zauważyć, że K_n pokolorowany jednym kolorem zawiera jednobarwną klikę K_n , a dwoma kolorami zawiera jednobarwną klikę K_2 . Jednocześnie K_{n-1} nie zawiera (w oczywisty sposób) jednobarwnej kliki K_n . Zatem $R(n, 2) = n$. Znajdowanie liczb Ramseya wyższych niż $K(3, 3)$ jest zadaniem bardzo trudnym [52].

- Ćw. 5.24 Skorzystaj z Tw. Halla.
- Ćw. 5.25 Ta naprawdę świetna sztuczka karciana opisana jest dokładnie w [65] (<http://courses.csail.mit.edu/6.042/spring10/cardTrick.pdf>), a także w ogólniejszym kontekście Tw. Halla w [66]. Na pierwszy rzut oka sztuczka wydaje się niemożliwa, bo ukazane 4 karty kodują $4! = 24$ możliwości, a pozostałych kart jest w talii jest $52 - 4 = 48$, czyli dwa razy więcej. Jednak asystent i iluzjonista wykorzystują w pewien subtelny sposób informację zawartą w piątej nieukazanej iluzji karcie, a także decydują, która karta ma być piąta. Przepis jest następujący: przynajmniej 2 karty spośród pięciu (na mocy zasadyszufladkowej) są tego samego koloru. Asystent umieszcza te dwie karty (w swojej głowie) na kolejne jak na rys. 7.12. Idąc po kolejne z wskazówkami zegara, jedna karta poprzedza drugą o odległość zawartą zawsze między 1 a 6. Asystent odkłada kartę będącą „z przodu” jako kartę do odgadnięcia i pokazuje iluzji jako pierwszą kartę tą będącą „z tyłu”, o walorze w . Iluzjonista zna już kolor! Następnie za pomocą pozostałych trzech kart asystent koduje odległość, pokazując karty w odpowiedniej kolejności. Wśród tych kart jest najstarsza (L), średnia (M) i najmłodsza (S). Sześciu permutacjom przyporządkowane są odległości, np.

$$LMS = 1, LSM = 2, MLS = 3, MSL = 4, SLM = 5, SML = 6.$$

Iluzjonista dodaje otrzymaną w ten sposób od asystenta liczbę do waloru w , korzystając z tego samego koła (jest to dodawanie modulo 13) i uzyskuje walor ukrytej karty.

W przykładzie z rys. 7.12 dwójka i dama mają ten sam kolor, dajmy na to pik. Dwójka poprzedza damę o 3, więc asystent pokazuje iluzję dama, a następnie pozostałe trzy karty w sekwencji MLS . Iluzjonista ogłasza: „Ukrytą przede mną kartą jest dwójka pik!”

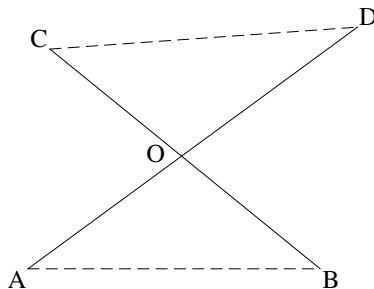


Rysunek 7.12: Dwójka poprzedza damę o 3 w kierunku zgodnym z ruchem wskazówek zegara (ćw. 5.25)

Ogólnie zauważmy, że za pomocą wariacji bez powtórzeń czterech kart można zakodować $52 \cdot 51 \cdot 50 \cdot 49$ sytuacji, czyli więcej niż liczba pięcioelementowych kombinacji kart: $\binom{52}{5} = 52 \cdot 51 \cdot 50 \cdot 49 \cdot \frac{48}{120}$. A priori możliwe jest więc skojarzenie kombinacji pięciu kart z wariacjami bez powtórzeń czterech kart. Bliższy związek problemu z Tw. Halla dyskutowany jest szczegółowo w [65].

7.6 Rozdział 6

- Ćw. 6.3. Zobacz np. <http://aturingmachine.com/index.php>.
- Ćw. 6.4. Wystarczy przyjąć odległości między każdą parą miast równą 1 i zadać pytanie, czy istnieje rozwiązanie problemu komiwojażera o długości $l \leq n + 1/2$.
- Ćw. 6.5. Gdyby istniało przecięcie, mielibyśmy sytuację jak na rys. 7.13, tzn. $|AB| + |CD| \leq |AO| + |BO| + |CO| + |DO| = |AD| + |CB|$. Trasa z odwijkanym przecięciem, zawierająca krawędzie oznaczone linią przerywaną, jest więc krótsza.



Rysunek 7.13: Skrzyżowanie krawędzi w problemie komiwojażera w zad. 6.5

Bibliografia

- [1] W. Broniowski, *Matematyka dyskretna. Wykłady z zadaniami dla studentów informatyki.* Wyd. I. Wydawnictwo UJK, Kielce 2015.
- [2] K. A. Ross, C. R. B. Write, *Matematyka dyskretna.* Wydawnictwo Naukowe PWN, Warszawa 2005.
- [3] R. L. Graham, D. E. Knuth, O. Patashnik, *Matematyka konkretna.* Wydawnictwo Naukowe PWN, Warszawa 2002.
- [4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Wprowadzenie do algorytmów.* Wydawnictwa Naukowo-Techniczne, Warszawa 2004.
- [5] A. Szepietowski, *Matematyka dyskretna.* Wydawnictwo Uniwersytetu Gdańskiego, Gdańsk 2004.
- [6] J. H. Conway, R. K. Guy, *Księga liczb.* Wydawnictwa Naukowo-Techniczne, Warszawa 2004.
- [7] W. Narkiewicz, *Teoria liczb.* Wydawnictwo Naukowe PWN, Warszawa 2003.
- [8] R. J. Wilson, *Wprowadzenie do teorii grafów,* wyd. 2. Wydawnictwo Naukowe PWN, Warszawa 2007.
- [9] V. Bryant, *Aspekty kombinatoryki.* Wydawnictwa Naukowo-Techniczne, Warszawa 1977.
- [10] W. Lipski, *Kombinatoryka dla programistów.* Wydawnictwa Naukowo-Techniczne, Warszawa 2004.
- [11] W. Lipski, W. Marek, *Analiza kombinatoryczna.* PWN, Warszawa 1986.
- [12] Z. Pałka, A. Ruciński, *Wykłady z kombinatoryki.* Wydawnictwa Naukowo-Techniczne, Warszawa 1998.
- [13] J. Jaworski, Z. Pałka, J. Szymański, *Matematyka dyskretna dla informatyków. I: elementy kombinatoryki.* UAM, Poznań 2011.
- [14] D. E. Knuth, *Sztuka programowania. T. 1, Algorytmy podstawowe.* Wydawnictwa Naukowo-Techniczne, Warszawa 2002.

- [15] D. E. Knuth, *Sztuka programowania*. T. 1, z. 1, *MMIX - komputer na nowe tysiąclecie*. Wydawnictwa Naukowo-Techniczne, Warszawa 2008.
- [16] D. E. Knuth, *Sztuka programowania*. T. 2, *Algorytmy seminumeryczne*. Wydawnictwa Naukowo-Techniczne, Warszawa 2002.
- [17] D. E. Knuth, *Sztuka programowania*. T. 3, *Sortowanie i wyszukiwanie*. Wydawnictwa Naukowo-Techniczne, Warszawa 2002.
- [18] D. E. Knuth, *Sztuka programowania*. T. 4, z. 2, *Generowanie wszystkich krotek i permutacji*. Wydawnictwa Naukowo-Techniczne, Warszawa 2007.
- [19] N. L. Biggs, *Discrete Mathematics*. Oxford University Press, Oxford 1989.
- [20] R. Garnier, J. Taylor, *Discrete Mathematics for New Technology*. IOP Publishing, Bristol 2004.
- [21] K. H. Rosen, *Discrete Mathematics and Its Applications*, 6th ed. McGraw Hill, Boston 2006.
- [22] G. Pólya, R. E. Tarjan, D. R. Woods, *Notes on Introductory Combinatorics*. Birkhäuser, Boston 1983.
- [23] J. Riordan, *An Introduction to Combinatorial Analysis*. Princeton University Press, Princeton 1978.
- [24] B. Bollobas, *Modern Graph Theory* (corr. ed.). Springer, New York 2013.
- [25] R. Diestel, *Graph Theory* (4th ed.). Springer, New York 2010.
- [26] R. Sedgewick, K. Wayne, *Algorithms*. Addison-Wesley, Boston 2011.
- [27] R. P. Grimaldi, B. V. Ramana, *Discrete and Combinatorial Mathematics, 5th edition*. Pearson Education, Harlow, Essex, W. Brytania 2006.
- [28] S. S. Epp, *Discrete Mathematics with Applications*. Brooks/Cole, Boston 2011.
- [29] A. Włoch, I. Włoch, *Matematyka dyskretna: podstawowe metody i algorytmy teorii grafów*. Oficyna Wydawnicza Politechniki Rzeszowskiej 2017.
- [30] P. Pusz, *Elementy matematyki dyskretnej*. Wydawnictwo Uniwersytetu Rzeszowskiego 2018.
- [31] W. Kordecki, A. Łyczkowska-Hanćkowiak, *Matematyka dyskretna dla informatyków*. Helion 2018.
- [32] S. Lipschutz, M. Lipson, M. L. Lipschutz, *2000 Solved Problems in Discrete Mathematics*
- [33] É. Lucas, *Récréations mathématiques*. Gauthier-Villars, Paryż 1891-1894. Reprint Albert Blanchard, Paryż 1960.

- [34] P. Singh, *The So-Called Fibonacci Numbers in Ancient and Medieval India*, Historia Mathematica, vol. 12, 1985, s. 229.
- [35] F. Leja, *Rachunek różniczkowy i całkowy*. PWN, Warszawa 1973.
- [36] Dan Brown, *Kod Leonarda da Vinci*. Albatros, Warszawa 2004.
- [37] M. Kraitchik, *Mathematical Recreations* (2nd ed.). Dover, New York 2006.
- [38] A. Karatsuba, Yu. Ofman, *Multiplication of Many-Digital Numbers by Automatic Computers*, Proceedings of the USSR Academy of Sciences, vol. 145, 1962, s. 293.
- [39] T. Leighton, *Notes on Better Master Theorems for Divide-and-Conquer Recurrences*. Notatki do wykładów, MIT, 1996 (<http://courses.csail.mit.edu/6.046/spring04/handouts/akrabazzi.pdf>).
- [40] S. Jeleński, *Lilavati*. Państwowe Zakłady Wydawnictw Szkolnych, Warszawa 1968.
- [41] K. P. Bogart, P. G. Doyle, *Non-sexist solution of the ménage problem*, Amer. Math. Monthly, vol. 93, 1986, s. 514.
- [42] B. Gleichgewicht, *Algebra*. PWN, Warszawa 1983.
- [43] A. I. Kostrikin, *Wstęp do algebra*. Cz. 1, *Podstawy algebra*. Wydawnictwo Naukowe PWN, Warszawa 2012.
- [44] F. C. Auluck, *On Some New Types of Partitions Associated with Generalized Ferrers Graphs*, Proceedings of the Cambridge Philosophical Society, vol. 47, 1951, s. 679.
- [45] N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences* (<http://www.research.att.com/~njas/sequences/Seis.html>).
- [46] R. E. Korf and A. Felner, *Recent Progress in Heuristic Search: A Case Study of the Four-Peg Towers of Hanoi Problem*, IJCAI, vol. 7, 2007, s. 2324.
- [47] V. Boltyanski, H. Martini, V. Soltan, V. P. Soltan, *Geometric Methods and Optimization Problems*. Springer, New York 1999.
- [48] C. Isenberg, *The Science of Soap Films and Soap Bubbles*. Dover, New York 1992.
- [49] D. J. Watts, *Six Degrees: The Science of a Connected Age*. Norton, New York 2003.
- [50] P. S. Dodds, R. Muhamad, D. J. Watts, *An Experimental Study of Search in Global Social Networks*, Science, vol. 301, 2003, s. 827.

- [51] F. M. Dong, K. M. Koh, K. L. Teo, *Chromatic polynomials and chromaticity of graphs*. World Scientific, Singapore 2005.
- [52] S. Radziszowski, *Small Ramsey Numbers*, Dynamic Surveys of the Electronic Journal of Combinatorics, vol. rev. 13, 2011.
- [53] J. E. Hopcroft, R. Motwani, J. D. Ullman, *Wprowadzenie do teorii automatów, języków i obliczeń*. PWN, Warszawa 2005.
- [54] D. C. Kozen, *Automata and Computability*. Springer, New York 1997.
- [55] S. Wolfram, *A New Kind of Science* (<http://www.wolframscience.com>).
- [56] R. Penrose, *Nowy umysł cesarza*. PWN, Warszawa 1995.
- [57] T. D. Gwiazda, *Algorytmy genetyczne. Kompendium*. T. 1 i 2. Wydawnictwo Naukowe PWN, Warszawa 2007.
- [58] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Bradford, Cambridge, Massachusetts 1992.
- [59] R. E. Bellman, S. E. Dreyfus, *Programowanie dynamiczne*. PWE, Warszawa 1967.
- [60] W. Stallings, *Kryptografia i bezpieczeństwo sieci komputerowych. Matematyka szyfrów i techniki kryptologii*, wyd. 5. Helion, Gliwice 2011.
- [61] J. Gribbin, *Kubity i kot Schrödingera. Od maszyny Turinga do komputerów kwantowych*. Wiedza i Życie - Orbity Nauki 2020.
- [62] M. Gimeno-Segovia, E. R. Johnston, N. Harrigan, *Komputer kwantowy. Programowanie, algorytmy, kod*. Helion 2020.
- [63] M. Le Bellac, *Wstęp do informatyki kwantowej*. Wydawnictwo Naukowe PWN, Warszawa 2012.
- [64] A. M. Yaglom, I. M. Yaglom, *Challenging Mathematical Problems with Elementary Solutions*, t. 1. Dover, New York 1987.
- [65] M. Kleber, *The Best Card Trick*, Mathematical Intelligencer, vol. 24, 2002, s. 9.
- [66] J. Jaszuńska, *Twierdzenie Halla o kojarzeniu małżeństw*, Matematyka-Społeczeństwo-Nauczanie, vol. 36, 2006, s. 17.

Skorowidz

- Akra, Mohamad A. 40
al-Chwarizmi, Muhammad ibn Musa 213
alfabet 215, 219, 221
 wejściowy 217
 wyjściowy 217
algorytm 213
 „196” 223
 Bellmana–Forda 161, 162
 brutalnej siły 142
 deterministyczny 214
 Dijkstry 159, 160
 Dinica 208
 Edmondsa-Karpa 208
 Fleury’ego 138
 Floyd–Warshalla 163
 Forda-Fulkersona 205
 genetyczny 226
 Hanoi 4
 heurystyczny 226
 iteracyjny 214
 Jarnika 156, 157
 kolorowania grafu 179, 180
 Kruskala 158
 najbliższego sąsiada 225
 probabilistyczny 214
 redukowalny 235
 rekursywny 214
 równoległy 214
 swatania 191
 szeregowy 214
 szybki 232
 wielomianowy 11
- wolny 232
wykładniczy 7
zachłanny 225
z powracaniem 142, 179, 191,
 225
Appel, Kenneth 181
asymptotyka 7
automat skończony
 deterministyczny 215, 216
 probabilistyczny 28
- Bayes, Thomas 95
Bazzi, Louay 40
bańki mydlane 173
Bell, Eric Temple 107
Bellman, Richard Ernest 161
błędzenie przypadkowe 27
błona mydlana 173
Bondy, John Adrian 140
Brooks, Rowland Leonard 179
bryły platońskie 142, 149, 150
butelka Kleina 183
- Cassini, Giovanni Domenico 16
Catalan, Eugène Charles 110
całka gaussowska 53
charakterystyka Eulera 151, 183
chiński listonosz 270
choroba wściekłych królów 96
Church, Alonzo 220
Chvátal, Václav 140
ciąg
 arytmetyczny 11

- Fibonacciego 11, 21, 69
- geometryczny 7
- postać ogólna 5
- rekurencyjny 5
- stały 12
- cięciwa grafu 153
- Cook, Stephen Arthur 46
- cykl
 - Eulera 136, 137, 236
 - Hamiltona 139, 143, 225, 236, 237
 - permutacji 102
 - singletowy 102
 - w grafie 135
- człon
 - jednorodny 23
 - niejednorodny 23
- deterministic finite automaton* zob.
 - automat skończony
 - deterministyczny
- DFA zob. automat skończony
 - deterministyczny
- diagram
 - Eulera 84
 - Ferrersa 115
 - sprzęzony 115
 - Venna 84, 85, 92
- digraf zob. graf skierowany
- Dijkstra, Edsger 156
- Dinic, Efim 208
- Dirac, Gabriel Andrew 140
- dobór naturalny 227
- domino 67, 72, 253
- dowód konstrukcyjny 4
- droga
 - Eulera 136
 - krytyczna 201
 - otwarta 135
 - prosta 135
 - w grafie 133
 - zamknięta 135
- drzewo 143
 - binarne 193
- pełne 194
- etykietowane 192
- pełne 193
- poszukiwań binarnych 194, 195
- regularne 193
- spinające 153
 - minimalne 156
- Steinera 166, 167
- z wyróżnionym korzeniem 192, 193
- DTM zob. Turinga maszyna
 - deterministyczna
- dwudziestościan foremny 150
- dwumian Newtona zob. wzór
 - dwumianowy Newtona, 61
- dwunastościan foremny 150
- dyskretność XIII
- dziecko
 - lewe 193
 - prawe 193
 - w drzewie 192
- dzielenie pizzy 42, 239
- Edmonds, Jack R. 208
- Efron, Bradley 98, 255
- energia potencjalna 168
- Erdős, Paul 175
- etykieta wierzchołka 194
- Euler, Leonhard 50, 148
- ϕ zob. złoty podział
- Fermat, Pierre de 47, 167
- Ferrers, Norman Macleod 115
- Fibonacci 11
- filotaksja 18
- Fleury, M. 138
- Floyd, Robert 163
- Ford, Lester Randolph, Jr. 161, 205
- Fulkerson, Delbert Ray 205
- full w pokerze 256
- fundamentalna redukcja 185
- funkcja
 - Γ Eulera 50, 52
 - całkowitoliczowa 39

- przejścia 215, 217, 219
tworząca 11, 72, 119
ciągu Fibonacciego 14
ciągu stałego 12
współczynników
dwumianowych 63
wykładnicza 109
wyjściowa 217
- gałąź drzewa 143
genealogia pszczół 17
głowica 218
Gödel, Kurt 221
Gombaud, Antoine 47, 98, 254
graf 129
 k -barwny 177
 k -chromatyczny 177
 k -spójny 134
 cykliczny 142, 178
 dualny 181, 182
 dwudzielny 142, 178
 pełny, $K_{n,m}$ 142
 eulerowski 136, 184
 hamiltonowski 139
 Hanoi 165
 nieskierowany 129
 Petersena 145, 146
 pełny, K_n 133
 planarny 146, 147
 platoński 142, 144
 półeulerowski 136
 półhamiltonowski 139
 połączeń internetowych 126
 prosty 133
 przyjaciół 126, 127
 pusty, N_n 133
 regularny 145, 146, 147
 skierowany 131
 spójny 134, 155
 z wagami 156
- grupa
 permutacji 101
 symetryczna 101
- gry hazardowe 47
- Guthrie, Francis 181
- Haken, Wolfgang 181
Hall, Philip 190
halting problem zob. problem zatrzymywania się
Hamilton, William Rowan 139
Hardy, Godfrey Harold 114
Hierholzer, Carl 137
Holland, John Henry 229
homeomorfizm grafów 146, 147
- iloczyn Cauchy'ego szeregów 64, 112
indukcja matematyczna 4
institut Claya XIV
interpretacja
 multiplikatywna NDTM 222
 probabilistyczna NDTM 222,
 223
izomorfizm grafów 145
- Jarnik, Vojtěch 156
język 216
- kalkulator Hewletta-Packarda 198
Karatsuba, Anatoli Aleksiejewicz 35
karta w pokerze 256
Karp, Richard Manning 208
Kirchhoff, Gustav Robert 203
klasa
 co-NP 237
 EXP 233
 LOG 233
 NP XIV, 167, 233, 235
 NPC 140, 224, 235
 P XIV, 233
 P-SPACE 237
 P-TIME 237
 problemów dopełniających 236
 QP 237
 klasyfikacja problemów decyzyjnych
 233
 Klein, Christian Felix 183
 klika 135
 Kod Leonarda da Vinci 19

- kojarzenie małżeństw 190, 208
kolor w pokerze 256
kolorowanie
 grafów 176, 185
 krawędziowe 185
 mapy na torusie 183
 wierzchołkowe 176, 177, 184
Kołmogorow, Andriej Nikołajewicz
 76
kombinacja 58, 121
 z powtórzeniami 62, 72, 121, 250
kombinatoryka 47
kompendium kombinatoryczne 121
komputer
 analogowy 167, 170, 173
 kwantowy 236, 237
 mydlany 173
komunikacja internetowa 159
kopiec Fibonacciego 19
Korona Kielce 87
korzeń drzewa 192
kości Efrona 98, 255
krawędzie sąsiednie 129
krawędź
 grafu 129, 225
 nasycona 204
 wielokrotna 133
Królewiec 127
króliki Fibonacciego 17
Kruskal, Joseph Bernard 158
kryptografia z kluczem publicznym
 236
Kuratowski, Kazimierz 147
Laplace, Pierre-Simon de 74
las 143
lemat
 o uściskach dloni 131, 149, 204
Leonardo z Pizy *zob.* Fibonacci
liczba
 chromatyczna 177
 Erdős-a 175
 palindromiczna 223
 partyjci 121
permutacji 49
przenicowana 223
usadowień na przyjęciu 93, 259
liczby
 Bella 107, 108
 Catalana 110, 113
 czworościenne 61
 dominowe 69
 Fibonacciego 11, 43, 68, 69
 Lucasa 42
 Ramseya 188, 189, 211
 Stirlinga 103
 drugiego rodzaju 105, 121
 pierwszego rodzaju 103
 trójkątne 10, 26, 61, 234
liść drzewa 143
lista
 infiksowa 196
 postfiksowa 196
 prefiksowa 195
listy incydencji 132
lodziarz z Pińczowa 224
Lucas, François Édouard Anatole 2,
 42, 270
łamigłówka Reve'a 166, 209
Łukasiewicz, Jan 197
macierz
 incydencji 130
 sąsiedztwa 130, 132
maksymalne skojarzenie 190, 208
mapa
 dwubarwna 184
 kartograficzna 182
 na płaszczyźnie 183
 na torusie 183
Markow, Andriej Andriejewicz 27
marszruta *zob.* droga
matematyka dyskretna XIII
matrioszka 122
Mealy, George H. 217
metoda
 „podstaw i rozwiń” 6, 10
 „zgadnij i sprawdź” 6

- bisekcji 232
brutalnej siły 225
najbliższego sąsiada *zob.*
 algorytm najbliższego
 sąsiada
metryka 156
Milgram, Stanley 173
Mises, Richard von 75
mnisi buddyjscy 7
model
 Mealy'ego 217
 Moora 217
 obliczeniowy 221
Moore, Edward F. 217
Morgan, Augustus de 181
mózg małpy 127
mysz w labiryncie 99
- napięcie powierzchniowe 173
nawigacja samochodowa 159
NDTM *zob.* Turinga maszyna
 niedeterministyczna
Newton, Sir Isaac 15
nieporządek 87
niezmiennik topologiczny 149, 151
notacja
 asymptotyczna 7
 odwrotna polska 198
 polska 198
- obwód grafu 135
odległość wierzchołków 135
Ofman, Yuri 35
ograniczenie
 dolne 8
 górnne 8
optymalizacja procesów 199
Ore, Oystein 140
- para w pokerze 257
paradoks
 hipochondryka 96
 urodzin 79, 80
parkiet 72, 253
- partitio numerorum* *zob.* partycja
 liczby
partycja liczby 113
Pascal, Blaise 61
Penrose, Roger 221
pentagram 18, 245
permutacja 48, 121
 z powtórzeniami 57, 67
PERT 199
pętla 132
Pick, Georg Alexander 167
pizza 42, 239
podgraf 135
podział
 koła cięciwami 44
 zbioru 106
poker 98, 255
pokrycie
 dominem 67, 70
 parkietem 253
Pólya, George 167
prawdopodobieństwo
 a posteriori 95
 a priori 95
 całkowite 29, 94
definicja
 aksjomatyczna 76
 częstościowa 75
 klasyczna 74
 dyskretne 74
 warunkowe 94
prawo
 ciągłości 204
 Kirchhoffa 203
 wielkich liczb 75
Prim, Robert Clay 156
problem
 „196” 223
 abstrakcyjny 230
 decyzyjny 232
 dopełniający 236
 gości przy okrągłym stole 55
kioskarza *zob.* rozmieniania

- kolorowania map 181
- komiwojażera 224, 234
 - niemieckiego 229
- konika szachowego 142, 209
- laptopa w taksówce 99
- nafciarza z Teksasu 166
- najkrótszej ścieżki 159
- nierozstrzygalny 230
- optymalizacyjny 232
- podziału godzin 176
- pomyłonych kapeluszy 87
- pomyłonych parasoli 87
- rozmieniania 123, 263
- rozstrzygalny 230
- sadowienia gości 91
 - zmodyfikowany 98
- szalików kibiców Korony 87, 98
- urodzin *zob.* paradoks urodzin
- zatrzymywania się 221
- proces
 - Markowa 27
- promocja karabinu „Beryl” 99
- przechowywanie danych 194
- przekaźnik 217
- przekrój
 - minimalny 205
 - sieci 204
- przekłamanie 97
- przepustowość
 - krawędzi 202
 - przekroju 204
 - rezydualna 205
 - sieci 202
- przepływ
 - maksymalny 204
 - w sieci 203
- przestrzeń
 - zdarzeń elementarnych 74
- przesunięcie indeksu sumowania 13
- przeszukiwanie
 - binarne 46
 - grafów 153
 - w głąb 153, 155
- wszerz 153, 155
- punkt Torricelliego 167
- rachunek prawdopodobieństwa 73
- Ramanujan, Srinivasa Aiyangar 114
- Ramsey, Frank Plumpton 189
- redukowalność algorytmów 235
- reguła
 - trapezów 51, 52
 - włączeń i wyłączeń 88
- rejestr stanów 218
- rekurencja 1, 5, 214
 - dziel i rządź 37
 - liniowa 22
 - jednorodna 21, 22
 - niejednorodna 23
 - uniwersalna 37
- relacja przejścia 221
- rezerwa czasowa wierzchołka 201
- rodzic w drzewie 192
- router 126
- rozdzieliczość problemu
 - optymalizacyjnego 232
- rozkład
 - liczby *zob.* partycja liczb
 - samosprzężony 115
 - liczby na czynniki pierwsze 236
- rozwiązywanie
 - ogólne 22
 - szczególne 24
- równanie charakterystyczne 19, 21
- ruina gracza 27, 246, 247
- rzucanie kostką 76
- scalanie problemu 37
- Schönhage, Arnold 38
- Sierpiński, Wacław 164
- sieć 202
 - rezydualna 205
 - zdarzeń 199, 200
- silnia 49, 51
- składanie permutacji 103
- słowo 216
- sortowanie przez scalanie 46

- stan XIII, 215, 217, 219, 221
akceptujący 28, 215, 219, 221
początkowy 215, 217, 219
- Steiner, Jakob 167
- Stirling, James 54
- stopień wierzchołka 130
wejściowy 202
wyjściowy 202
- Strassen, Volker 38
- street w pokerze 256
- symbol
 Ω 8
 Θ 8
 \sim 7, 8
 \mathcal{O} 8, 233
 \circ 9
Newtona 59
Pochhammera 57
- szarlotka Babuni 199
- szereg
geometryczny 12, 14
Taylora 63
- sześć stopni oddalenia 174
- szylkcie mnożenie 35
- szłyfr RSA 236
- ścieżka *zob.* droga
- ślad macierzy 208
- średnica grafu 135
- tańce w kółku 103
- taśma 218
- test ciążowy 97
- Tezeusz 153
- Toom, Andriej 45
- topologia
drzewa Steinera 169
powierzchni 151
- Torricelli, Evangelista 167
- torus 150, 151
- tożsamość
Cassiniego 16
Cauchy'ego 64
- triangulacja wielokąta 110
- trójkąt w pokerze 256
- trójkąt
Bella 108
liczb Stirlinga
drugiego rodzaju 107
pierwszego rodzaju 105
- liczb szalikowych 91
- liczby partycji 117
na składniki różne 119
- Pascala 60, 61, 69
- Sierpińskiego 164, 270
- Turing, Alan 214
- Turinga
maszyna 214, 218
deterministyczna 219
niedeterministyczna 221
uniwersalna 220
- Turinga-Churcha hipoteza 220
- turniej szachowy 98
- twierdzenie
Bayesa 95
Bondy'ego-Chvatala 140
Diraca 140
Eulera 136
Halla 190
Kuratowskiego 147
o czterech kolorach 181
o kolorowaniu map 182
o maksymalnym przepływie
i minimalnym przekroju 205
Oregonego 140
- ujście 202
- układanie
domina 67, 72, 253
parkietu 67, 72, 253
- ułamki proste 14
- UTM *zob.* Turinga maszyna
uniwersalna
- von Neumann, John 46
- waga krawędziowa grafu 156
- wariacja
bez powtórzeń 57, 121

- z powtórzeniami 55, 58, 121
Warshall, Stephen 163
warunek
 ciągły 203
 kąta
 dla drzewa Steinera 173
 dla problemu Fermata 169
 wielomianowości 40, 41
warunki początkowe 19
węzeł gałęzi drzewa 192
wiek Wszechświata 7, 231
wielkość przepływu 204
wielomian chromatyczny 184
wielościan foremny 149
wierzchołek
 grafu 129, 225
 Steinera 167
 wiszący *zob.* liść drzewa
wierzchołki sąsiednie 129
Wieże Hanoi 1, 164, 234
współczynnik
 dwumianowy 63
 wielomianowy 65
wysokość drzewa 192
- wzór
 Dobińskiego 110
 dwumianowy Newtona 15, 61
 Stirlinga 54
 wielościanu Eulera 148
 ogólny 151
- zasada
 szufladkowa Dirichleta 65, 72,
 252
 włączania i wyłączania 84, 86,
 92
zdarzenia niezależne 78
zdarzenie 74
 elementarne 74
zliczanie
 grafów 152
złoty
 podział 15
 odcinka 18
 prostokąt 18
złożoność problemów decyzyjnych
 233
źródło 202

Przykłady w języku Python

18 sie 2021

Spis treści

1 Rekurencja	3
1.1 Wieże Hanoi	3
1.2 Ciąg Fibonacciego	8
1.3 Metoda równana charakterystycznego	9
1.4 Błędzenia przypadkowe	12
1.5 Silnia jako rekurencja	17
1.6 Rekurencja w dwóch wymiarach	19
1.7 Zadania	23
2 Grafy	25
2.1 Rozgrzewka	26
2.2 Mosty królewskie	30
2.3 Kalejdoskop grafów	35
2.4 Badanie grafów	37
2.5 Algorytmy na grafach	40
2.6 Grafy zewnętrznych danych	53
2.7 Kolorowanie grafów	58
2.8 Przepływ w sieci	59
2.9 Drzewo Steinera	63
2.10 Zadania	65

Niniejsza wykonywalna książka stanowi integralną część podręcznika **Matematyka dyskretna, wyd. II: wykłady z przykładami w języku Python** i zawiera proste programy ilustrujące wybrane problemy z rekurencji i teorii grafów.

Odbońniki

Odbońniki do problemów wyjaśnianych w pierwszej części książki oznaczone są klamrami i wytluszczoną czcionką.

Wymagania wstępne

Czytelnik powinien być zaznajomiony z elementarnymi podstawami języka **Python** oraz ze środowiskiem Jupyter.

Jak puszczać programy

Podstawową zaletą powstały niedawno wykonywalnych książek jest możliwość puszczania zawartych w nich programów bez kopiowania, ściągania, czy instalacji. Możliwa jest przy tym dowolna modyfikacja, rozszerzenia, „zabawa” z parametrami kodu, a także przesyłanie sugestii ulepszeń do autora. Te cechy sprawiają, że wykonywalne książki są świetną pomocą dydaktyczną dla studentów informatyki.

Aby wykonać programy należy pójść do strony

<https://bronwojtek.github.io/matematyka-dyskretna-python/docs/index.html>.

Po wybraniu konkretnego rozdziału w menu po lewej stronie (Rekurencja lub Grafy), w prawym górnym rogu ekranu należy kliknąć ikonę rakietki i wybrać Binder. Po pewnym, za pierwszym razem dość długim czasie, w przeglądarce otworzy się notebook Jupytera, który jest do naszej dyspozycji. Ta najwygodniejsza metoda wykonuje program w chmurze obliczeniowej.

Alternatywnie, klikając na ikonę strzałki w dół, można pobrać kod i wykonać go lokalnie, otwierając jako notebook Jupytera np. w środowisku **anaconda**. Oczywiście, metoda lokalna wymaga wsześniejszej instalacji odpowiedniego oprogramowania.

Linki

- Jupyter Book: <https://bronwojtek.github.io/matematyka-dyskretna-python/docs/index.html>
 - pdf całej książki i programy: www.ifj.edu.pl/~broniows/md lub www.ujk.edu.pl/~broniows/md
-

Built with [Jupyter Book 2.0](#) tool set, as part of the [ExecutableBookProject](#).

ROZDZIAŁ 1

Rekurencja

Standardowe biblioteki użyte w programie:

```
import numpy      # numeryka
import math       # funkcje matematyczne
import cmath      # liczby zespolone
import random    # liczby losowe
import time       # pomiar czasu
import statistics as st      # statystyka
import matplotlib.pyplot as plt # grafika
```

1.1 Wieże Hanoi

Na początek klasyczny przykład algorytmu rekurencyjnego z **rozdz. 1.1**.

Etykiety krążków **disc** zdefiniowane są jako krotka („A”, „B”, „C”, „D”, ...), gdzie potrzebujemy co najmniej tyle etykiet, ile wynosi używana później liczba krążków.

```
disc=("A", "B", "C", "D", "E") # użyjemy co najwyżej pięciu krążków
```

Poniższa funkcja realizuje algorytm rekureencyjny [1.1] (który tutaj wypisuje tylko kolejne instrukcje postępowania, tj. kroki algorytmu). Program **han** woła (dwakroć) sam siebie, zawsze z mniejszą o 1 liczbą krążków.

Przykłady w języku Python

```
def han(n, init, fin, inter):  
  
    # n - liczba krążków  
    # init - pręt, na którym są początkowo krążki  
    # fin - pręt, na którym mają się na koniec znaleźć krążki  
    # inter - pręt pośredni  
  
    if n > 0:    # wykonaj, jeśli są krążki  
  
        han(n-1, init, inter, fin)  # przenieś n-1 krążków z  
        ↪init na inter  
  
        print("Przenieś krążek", disc[n-1], "z pręta", init, "na"  
        ↪pręt", fin)  
        # przenieś krążek n z pręta init na fin (wypisz  
        ↪instrukcję)  
  
        han(n-1, inter, fin, init)  # przenieś n-1 krążków z  
        ↪inter na fin
```

```
# wykonanie dla trzech krążków  
han(3, "1", "3", "2")
```

```
Przenieś krążek A z pręta 1 na pręt 3  
Przenieś krążek B z pręta 1 na pręt 2  
Przenieś krążek A z pręta 3 na pręt 2  
Przenieś krążek C z pręta 1 na pręt 3  
Przenieś krążek A z pręta 2 na pręt 1  
Przenieś krążek B z pręta 2 na pręt 3  
Przenieś krążek A z pręta 1 na pręt 3
```

Powyżej możemy prześledzić dokładnie „historyjkę” z [Rys. 1.2].

```
# wykonanie dla czterech krążków  
han(4, "1", "2", "3")
```

```
Przenieś krążek A z pręta 1 na pręt 3  
Przenieś krążek B z pręta 1 na pręt 2  
Przenieś krążek A z pręta 3 na pręt 2  
Przenieś krążek C z pręta 1 na pręt 3  
Przenieś krążek A z pręta 2 na pręt 1  
Przenieś krążek B z pręta 2 na pręt 3  
Przenieś krążek A z pręta 1 na pręt 3  
Przenieś krążek D z pręta 1 na pręt 2  
Przenieś krążek A z pręta 3 na pręt 2  
Przenieś krążek B z pręta 3 na pręt 1  
Przenieś krążek A z pręta 2 na pręt 1  
Przenieś krążek C z pręta 3 na pręt 2  
Przenieś krążek A z pręta 1 na pręt 3
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
Przenieś krążek B z pręta 1 na pręt 2
Przenieś krążek A z pręta 3 na pręt 2
```

Poniżej bardziej zaawansowana wersja, gdzie śledzimy kolejne stany (tj. ułożenia krążków) na prętach 1, 2, 3. Stany określone są jako łańcuchy znaków A, B, C, ... etykietujących krążki, tak jak w **rozdz. 1.1**. Liczba kroków **step** określona jest jako **global**, ponieważ inicjalizujemy ją poza funkcją.

```
def han2(n, init, fin, inter):

    # n - liczba krążków
    # init   - pręt, na którym są początkowo krążki
    # fin    - pręt, na którym mają się na koniec znaleźć krążki
    # inter  - pręt pośredni

    global step # liczba kroków

    if n > 0:    # zrób, jeśli są krążki
        han2(n-1, init, inter, fin) # przenieś n-1 krążków z
        ↪init na inter

        fin.append(init.pop())
        step+=1 # licz kroki
        print('krok',step,":")
        print('1:',''.join(one))
        print('2:',''.join(two))
        print('3:',''.join(three))

        han2(n-1, inter, fin, init) # przenieś n-1 krążków z
        ↪inter na fin
```

```
# Stan początkowy: wszystkie trzy krążki na pręcie "1",
# od największego C do najmniejszego A
one  = ['C', 'B', 'A']
two  = []
three = []

step=0

print('krok', step, ":")
print('1:', ''.join(one))
print('2:', ''.join(two))
print('3:', ''.join(three))

han2(3, one, three, two)
```

```
krok 0 :
1: CBA
```

(ciąg dalszy na następnej stronie)

Przykłady w języku Python

(kontynuacja poprzedniej strony)

```
2:  
3:  
krok 1 :  
1: CB  
2:  
3: A  
krok 2 :  
1: C  
2: B  
3: A  
krok 3 :  
1: C  
2: BA  
3:  
krok 4 :  
1:  
2: BA  
3: C  
krok 5 :  
1: A  
2: B  
3: C  
krok 6 :  
1: A  
2:  
3: CB  
krok 7 :  
1:  
2:  
3: CBA
```

Jeśli interesuje nas tylko liczba kroków algorytmu Hanoi, implementujemy rekurencję [1.1] w następujący sposób:

```
# Ciąg Hanoi  
def hanoi(n):  
    if n == 1:  
        return 1  
    else:  
        return 2*hanoi(n-1)+1
```

Możemy teraz sprawdzić równanie (1.2):

```
for n in range(1,10): print(n, hanoi(n), 2**n-1)
```

```
1 1 1  
2 3 3  
3 7 7
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
4 15 15
5 31 31
6 63 63
7 127 127
8 255 255
9 511 511
```

Pamiętacie mnichów buddyjskich? Przenoszą 64 złote krążki między diamentowymi prętami.

```
hanoi(64)
```

```
18446744073709551615
```

```
2**64-1
```

```
18446744073709551615
```

```
# zakładamy, że jeden ruch trwa 5 sekund i przeliczamy czas na
# lata
x = (2**64-1)*5/60/60/24/365

# format liczby z wykładnikiem
y = '%.1E' % x
print("czas pracy mnichów:",y,"lat")

w=13.7*10**9 # wiek Wszechświata
print(round(x/w),"razy dłużej od wieku Wszechświata")
```

```
czas pracy mnichów: 2.9E+12 lat
213 razy dłużej od wieku Wszechświata
```

Rekurencja z zadania [1.10] posiada n^2 w miejscu 1, co prowadzi do oczywistej modyfikacji:

```
# ciąg Hanoi, gdzie liczymy przeniesioną masę krążków
# krążek n ma masę  $n^{**}2$ 
def hanoi_m(n):
    if n == 1:
        return 1
    else:
        return 2*hanoi_m(n-1)+n**2 # tutaj modyfikacja
```

```
[hanoi_m(n) for n in range(1,11)] # tablica 10 pierwszych wyrazów
```

```
[1, 6, 21, 58, 141, 318, 685, 1434, 2949, 5998]
```

1.2 Ciąg Fibonacciego

Ciąg Fibonacciego [1.8] programujemy analogicznie do powyższych przykładów:

```
def fib(n):
    if n == 1:
        return 1
    elif n==2:
        return 1
    else:
        return fib(n-1)+fib(n-2) # suma dwóch poprzednich wyrazów
```

```
fib(9)
```

```
34
```

```
[fib(i) for i in range(1, 11)]
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Kolejne ilorazy F_{i+1}/F_i , zaokrąglone do 6 cyfr znaczących:

```
[round(fib(i+1)/fib(i),6) for i in range(1,16)]
```

```
[1.0,
 2.0,
 1.5,
 1.666667,
 1.6,
 1.625,
 1.615385,
 1.619048,
 1.617647,
 1.618182,
 1.617978,
 1.618056,
 1.618026,
 1.618037,
 1.618033]
```

Jak wiemy z równ. [1.26], powyższe ilorazy dążą do Złotego Podziału ϕ [1.27]:

```
phi=(1+math.sqrt(5))/2.
print(phi)
```

```
1.618033988749895
```

Jeśli chcemy utworzyć tablicę kolejnych wyrazów ciągu Fibonacciego, postępujemy następująco:

```
fi=[]          # pusta tablica
fi.append(1)    # pierwszy wyraz
fi.append(1)    # drugi wyraz

for i in range(2,20):
    fi.append(fi[i-2]+fi[i-1]) # kolejne wyrazy

print(fi)
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765]
```

1.3 Metoda równana charakterystycznego

(dla rekurencji liniowej rzędu 2, **rozdz. 1.5**)

Zakładamy, że rekurencja dla ciągu f_n ma postać

$af_n + bf_{n-1} + cf_{n-2} = 0$ (przy czym stała a jest różna od 0),

np. dla ciągu Fibonacciego $a = 1, b = -1, c = -1$. Rekurencja rzędu 2 wymaga dwóch warunków „początkowych”, znamy zatem dwa różne wyrazy ciągu, n i m . Słowo „początkowych” ujęte jest w cudzysłów, bowiem niekoniecznie $n = 1, m = 2$, jak w przypadku ciągu Fibonacciego. Wskaźniki n i $m \neq n$ mogą być dowolne i wynikają z natury konkretnego problemu.

Poniższe pomocnicze funkcje obliczają wyróżnik równania kwadratowego Δ oraz jego pierwiastki:

```
# delta dla równania kwadratowego
def de(a, b, c):
    return b*b-4*a*c

# pierwiastki równania kwadratowego
def x1x2(a, b, c):
    d=de(a,b,c)    # wyróżnik
    if d>0:        # dla d>0 pierwiastki są rzeczywiste
        return [(-b+math.sqrt(d))/(2*a), (-b-math.sqrt(d))/(2*a)]
    if d<0:        # dla d<0 pierwiastki są zespolone
```

(ciąg dalszy na następnej stronie)

Przykłady w języku Python

(kontynuacja poprzedniej strony)

```
return [(-b+cmath.sqrt(d))/(2*a), (-b-cmath.sqrt(d))/  
        ↪(2*a)] # biblioteka cmath  
if d==0:      # dla d=0 jest jeden podwójny pierwiastek  
    ↪rzeczywisty  
    return -b/(2*a)
```

Oto poszczególne przypadki (symbol **j** oznacza w bibliotece **cmath** jednostkę urojoną):

```
print(de(1,3,1), x1x2(1,3,1))  
print(de(1,2,2), x1x2(1,2,2))  
print(de(1,2,1), x1x2(1,2,1))
```

```
5 [-0.3819660112501051, -2.618033988749895]  
-4 [(-1+1j), (-1-1j)]  
0 -1.0
```

Liczby zespolone wprowadzamy używając symbolu **j**, np. w następujący sposób:

```
1j**2
```

```
(-1+0j)
```

Algorytm [1.2] programujemy w następujący sposób:

```
def sol_rec(a,b,c,n,an,m,am,i):  
    """  
        Rozwiąż rekurencję liniową rzędu 2  
  
        input:  
        a, b, c - parametry (a różne od 0)  
        n - wskaźnik jednego znanego wyrazu  
        an - jego wartość  
        m - wskaźnik drugiego znanego wyrazu (n różne od m)  
        am - jego wartość  
        i - wskaźnik szukanego wyrazu  
  
        output:  
        wartość wyrazu i  
    """  
  
    if de(a,b,c) != 0:                      # dwa różne pierwiastki (delta  
        ↪różna od 0)  
        [x1, x2]=x1x2(a, b, c)  
  
        # ogólne rozwiązanie na stałe c1 i c2  
        c1=(an*x2**m - am*x2**n)/(x1**n*x2**m - x1**m*x2**n)  
        c2=(-an*x1**m + am*x1**n)/(x1**n*x2**m - x1**m*x2**n)
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```

return c1*x1**i+c2*x2**i # ogólna postać wyrazu i

else: # jeden pierwiastek (delta=0) ←
    ↵
    x=x1x2(a, b, c)

    # ogólne rozwiązanie na stałe c1 i c2
    c1=((am*n)/x**m - (an*m)/x**n)/(n - m)
    c2=(am/x**m - an/x**n)/(m - n)

return (c1+c2*i)**x**i # ogólna postać wyrazu i

```

Poniżej sprawdzenie dla ciągu Fibonacciego. Ponieważ wiemy, że wyrazy ciągu są liczbami całkowitymi, w celu uzyskania czytelniejszego wydruku stosujemy funkcje **real** oraz **int**.

```
[int(sol_rec(1,-1,-1,1,1,2,1,i)).real for i in range(1,11)]
```

```
[0, 0, 1, 2, 4, 8, 13, 21, 34, 55]
```

Inne przykłady:

```

def round_complex(x): # zaokrąglanie liczb zespolonych
    return complex(round(x.real),round(x.imag))

```

```
[int(round_complex(sol_rec(1,-1,1,1,1,2,1,i)).real) for i in range(1,11)]
```

```
[1, 1, 0, -1, -1, 0, 1, 1, 0, -1]
```

```
[round(sol_rec(1,2,1,1,1,2,1,i)) for i in range(1,11)]
```

```
[1, 1, -3, 5, -7, 9, -11, 13, -15, 17]
```

```
[int(round_complex(sol_rec(1,-3,3,1,1,2,1,i)).real) for i in range(1,17)]
```

```
[1, 1, 0, -3, -9, -18, -27, -27, 0, 81, 243, 486, 729, 729, 0, -2187]
```

1.4 Błądzenia przypadkowe

1.4.1 Liczby pseudolosowe

Symulowanie procesów stochastycznych wymaga zastosowania liczb pseudolosowych, wbudowanych we wszystkie zaawansowane języki programowania. Zaczynamy od przykładu kostki do gry (3.1), używając biblioteki **random**.

```
# kostka do gry: losowa liczba naturalna w zakresie od 1 do 6
random.randint(1, 6)
```

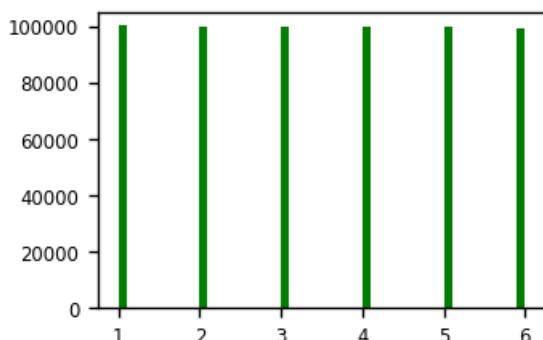
```
6
```

Rzucamy 600000 razy i tworzymy histogram (por. **rys. 3.1**):

```
# zbieramy wyniki kolejnych rzutów w tablicy
t=[random.randint(1,6) for i in range(600000)]

plt.figure(figsize=(3,2), dpi=120) #←
    ↪rozmiar grafiki
plt.tick_params(axis='both', which='major', labelsize=7) #←
    ↪rozmiar znaczników osi

plt.hist(t, bins=50, facecolor='green')
plt.show()
```



Widzimy, że każde oczko wypada po około 100000, czyli kostka jest dobra (niesfałszowana!).

1.4.2 Ruina gracza

W problemie ruiny gracza (**rozdz. 1.7**) losowanie odbywa się „sfałszowaną” monetą, gdzie prawdopodobieństwo uzyskania orła wynosi p , a reszki $q = 1 - p$. Poniżej tworzymy generator liczb losowych, który zwraca 1 z prawdopodobieństwem p oraz -1 z prawdopodobieństwem q . Uzyskanie 1 oznacza przejście do stanu posiadania większego o 1, a -1 do stanu posiadania mniejszego o 1 (zob. **rys. 1.8**).

```
def coin(p):
    # zwrócić 1 z prawdopodobieństwem p lub -1 z prawdopodobieństwem q=1-p

    if random.uniform(0,1)<p: # gdy liczba losowa rozłożona jednorodnie
        # w przedziale (0,1) jest mniejsza od p
        # (tj. prawdopodobieństwo tegoż wynosi p)
        return 1
    else:
        # zwrócić 1
        # w przeciwnym razie
        # zwrócić -1
```

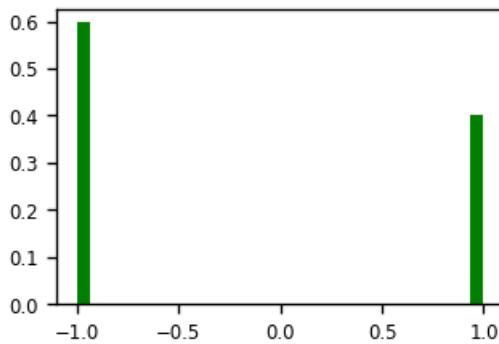
Sprawdzamy teraz działanie funkcji **coin** (analogicznie jak dla kostki do gry powyżej). Parametr **weights** zmienia wysokość słupków tak, aby odpowiadała ona względnej częstości uzyskania 1 lub -1.

```
p=0.4      # prawdopodobieństwo orła
nu=100000  # liczba rzutów
t=[coin(p) for k in range(1,nu)]

plt.figure(figsize=(3,2),dpi=120)          #
# rozmiar grafiki
plt.tick_params(axis='both', which='major', labelsize=7) #
# rozmiar znaczników osi

plt.hist(t, bins=30, facecolor='green', weights=1/nu*numpy.ones_like(t))
plt.show()
```

Przykłady w języku Python



Pełna symulacja ruiny gracza jest następująca:

```
time0 = time.time()      # czas startu do pomiaru czasu wykonania  
                        # programu

n = 40000            # maksymalna liczba kroków (tj. pojedynczych  
                      # gier dla jednego gracza)
p = 0.495            # prawdopodobieństwo wygrania w pojedynczej grze  
                      # (uzyskania orła)
q = 1-p              # prawdopodobieństwo przegrana w pojedynczej  
                      # grze

players    = 200     # całkowita liczba graczy
players_b = 0       # aktualna liczba graczy, którzy rozbili bank

start = 100          # początkowa liczba monet u każdego z graczy
fin   = 200          # liczba monet u gracza, przy której następuje  
                      # rozbicie banku

x = list(range(n))  # współrzędne x (nr kroku, tj. kolejnej  
                      # pojedynczej gry)

# ustawienia rysunku
plt.figure(figsize=(20,3))                                # rozmiar
plt.title("Ruina gracza, p=" + str(p), fontsize=24)        # tytuł
plt.xlim(0,n-1)                                         # zakresy osi
plt.ylim(0,fin)                                         # etykiety osi
plt.xlabel('krok', fontsize=24)                           # etykiety osi
plt.ylabel('stan posiadania', fontsize=24)
plt.tick_params(axis='both', which='major', labelsize=16)  # rozmiar znaczników osi

# losowe kolory RGB dla poszczególnych graczy
col = [(random.uniform(0,1),random.uniform(0,1),random.uniform(0,
                  1))]
for _ in range(players)]
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```

# czasy gry dla wszystkich, tych co rozbili bank i tych, co się
# spłukali
ti=[]
ti_b=[]
ti_0=[]

for k in range(players): # pętla po gracach

    y = numpy.zeros(n)      # inicjalizacja macierze
    i=0                     # początek gry, krok 0
    y[0]=start              # stan posiadania w kroku 0

    while i < n-1:
        i+=1                 # kolejny krok
        y[i]=y[i-1] + coin(p) # posiadanie losowo zwiększa się
        # lub zmniejsza o 1
        if y[i]==0:           # jeśli spłukany po i grach
            ti_0.append(i)    # zapisz liczbę kroków (czas gry)
            break               # wyjdź
        if y[i]==fin:          # jeśli rozbił bank po i grach
            players_b+=1      # zwiększa o 1 liczbę graczy,
        # którzy rozbili bank
            ti_b.append(i)    # zapisz liczbę kroków (czas gry)
            break               # wyjdź

        plt.scatter(x, y, s=0.01, label=str(k+1)) # zrób wykres dla
        # gracza k
        ti.append(i)             # zapisz czas gry

    lw= max(ti)                # najdłuższy czas gry

    plt.xlim(0,1.1*lw) # zakres osi x o 10% dłuższy od najdłuższego
    # czasu gry

    # wstawka tekstowa do rysunku
    plt.text(x=0.85*lw, y=80, s="spłukanych "+str(len(ti_0)),
    # fontsize=24, color='red')
    plt.text(x=0.85*lw, y=120, s="rozbilo bank "+str(len(ti_b)),
    # fontsize=24, color='blue')

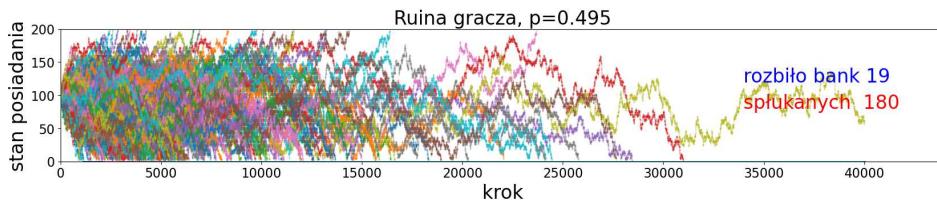
    # pomiar czasu umożliwia oszacowanie jak długo potrwają
    # obliczenia przy wydłużeniu danych
    # w naszym przypadku rośnie on liniowo z liczbą graczy
    time1 = time.time()
    print("czas wykonania symulacji:", round(time1 - time0, 1),"s")

plt.show()

```

Przykłady w języku Python

```
czas wykonania symulacji: 25.3 s
```



Możemy porównać oszacowane z powyższej symulacji prawdopodobienstwo rozbicia banku (równe stosunkowi liczby graczy, którzy rozbili bank, do wszystkich graczy), z dokładnym wzorem na prawdopodobieństwo [1.45]:

```
print("oszacowane z symulacji prawd. rozbicia banku =",  
    ↪round(players_b/players,2),  
    "\npowinno być (dla b. dużej liczby graczy)",round((1-(q/  
    ↪p)**start)/(1-(q/p)**fin),2))
```

```
oszacowane z symulacji prawd. rozbicia banku = 0.1  
powinno być (dla b. dużej liczby graczy) 0.12
```

Podobnie, oszacowany średni czas gry porównyjmy do wzoru (1.48):

```
print("średni czas gry:",round(st.mean(ti),1),", powinno być  
    ↪(dla b. dużej liczby graczy) ",  
    ↪round((-fin*(q/p)**start+start*(q/p)**fin)/(1-(q/p)**fin)/  
    ↪(p-q),1))  
  
print("średni czas gry dla tych, co rozbili bank:",round(st.  
    ↪mean(ti_b),1))  
  
print("średni czas gry dla tych, co się spłukali:",round(st.  
    ↪mean(ti_0),1))
```

```
średni czas gry: 7524.3 , powinno być (dla b. dużej liczby  
    ↪graczy) 7429.5  
średni czas gry dla tych, co rozbili bank: 8864.4  
średni czas gry dla tych, co się spłukali: 7202.4
```

Widzimy, że średni czas gry dla graczy, którzy rozbili bank, jest dłuższy niż dla tych, co się spłukali (co jest intuicyjne dla $p < 0.5$).

Ciekawe jest popatrzyć na rozkład czasów gry dla poszczególnych graczy (por. **rys. [1.11]):

```
plt.figure(figsize=(3,2),dpi=120)  
    ↪rozmiar grafiki
```

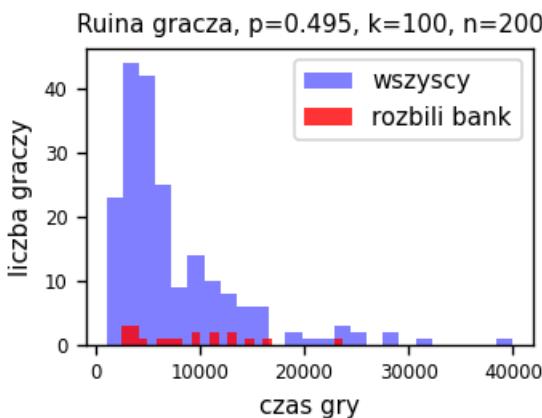
(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```

num_bins = 25    # liczba przedziałów histogramu
plt.hist(ti, num_bins, facecolor='blue', alpha=0.5,label="wszyscy"
         ↵")
plt.hist(ti_b, num_bins, facecolor='red', alpha=0.8,label=
         ↵"rozbili bank")
plt.title("Ruina gracza, p="+str(p)+", k="+str(start)+", n=
         ↵"+str(fin), fontsize=9)
plt.xlabel('czas gry', fontsize=9)
plt.ylabel('liczba graczy', fontsize=9)
plt.tick_params(axis='both', which='major', labelsize=7) #_
         ↵rozmiar znaczników osi
plt.legend(prop={'size':9})                                #_
         ↵legenda
plt.show()

```



1.5 Silnia jako rekurencja

Obliczamy rekurencyjnie silnię:

```

def fact(n):
    if n == 0:
        return 1
    else:
        return n*fact(n-1)

```

```
fact(7)
```

```
5040
```

Przykłady w języku Python

```
# biblioteki matematyczne Pythona mają wbudowaną silnię, np.  
math.factorial(7)
```

```
5040
```

Sprawdzimy teraz, jak działa wzór Stirlinga z poprawką [2.9] (bierzemy tylko jeden człon poprawki $1/(12n)$):

```
def stirling(n):  
    return math.sqrt(2*math.pi*n)*(n/math.e)**n*(1+1/(12*n))  
  
for n in range(1,20):  
    print(n,math.factorial(n),  
          round((stirling(n)-math.factorial(n))/math.  
          factorial(n)*100,3), "%")
```

```
1 1 -0.102 %  
2 2 -0.052 %  
3 6 -0.028 %  
4 24 -0.017 %  
5 120 -0.012 %  
6 720 -0.008 %  
7 5040 -0.006 %  
8 40320 -0.005 %  
9 362880 -0.004 %  
10 3628800 -0.003 %  
11 39916800 -0.003 %  
12 479001600 -0.002 %  
13 6227020800 -0.002 %  
14 87178291200 -0.002 %  
15 1307674368000 -0.001 %  
16 20922789888000 -0.001 %  
17 355687428096000 -0.001 %  
18 6402373705728000 -0.001 %  
19 121645100408832000 -0.001 %
```

Widzimy, że przybliżenie działa bardzo dokładnie, do ułamka procenta, nawet dla małych n . Uwzględnienie kolejnych członów we wzorze [2.9] jeszcze bardziej ulepsza wynik.

1.6 Rekurencja w dwóch wymiarach

1.6.1 Symbol Newtona

Skonstruujemy trójkąt Pascala, rys. 2.7 (pamiętamy, że „+” łączy listy):

```
newton=[] # inicjalizacja pustej tablicy

# dodanie [1] jako elementu 0 tablicy
newton.append([1])

# zrobimy 12 wierszy
for n in range(1,12):
    # wiemy, że po bokach są jedynki, stąd [1]+ i +[1]
    # w środku pętla po k daje (n-1) elementów
    # każdy jest sumą elemntów z wiersza wyżej w pozycji k-1
    # oraz k
    newton.append([1]+[newton[n-1][k-1]+newton[n-1][k] for k in
    range(1,n)]+[1])
```

Tablica **newton** jest dwuwymiarowa:

```
newton
```

```
[[1],
 [1, 1],
 [1, 2, 1],
 [1, 3, 3, 1],
 [1, 4, 6, 4, 1],
 [1, 5, 10, 10, 5, 1],
 [1, 6, 15, 20, 15, 6, 1],
 [1, 7, 21, 35, 35, 21, 7, 1],
 [1, 8, 28, 56, 70, 56, 28, 8, 1],
 [1, 9, 36, 84, 126, 126, 84, 36, 9, 1],
 [1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1],
 [1, 11, 55, 165, 330, 462, 462, 330, 165, 55, 11, 1]]
```

Ładniejszy wydruk uzyskujemy poprzez sformatowane wypisywanie:

```
for n in range(12):
    row_format = "{:>4}" * (n+1)
    print((12-n)*" ",row_format.format(*newton[n]))
```

```

          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

(ciąg dalszy na następnej stronie)

Przykłady w języku Python

(kontynuacja poprzedniej strony)

1	5	10	10	5	1						
1	6	15	20	15	6	1					
1	7	21	35	35	21	7	1				
1	8	28	56	70	56	28	8	1			
1	9	36	84	126	126	84	36	9	1		
1	10	45	120	210	252	210	120	45	10	1	
1	11	55	165	330	462	462	330	165	55	11	1

Sprawdzamy, że suma elementów w wierszu trójkąta Pascala daje 2^n , zgodnie ze wzorem [2.14]:

```
for n in range(12):
    print(sum(newton[n]), 2**n)
```

1 1
2 2
4 4
8 8
16 16
32 32
64 64
128 128
256 256
512 512
1024 1024
2048 2048

Podobnie, suma elementów w wierszu trójkąta Pascala mnożonych naprzemiennie przez -1 daje 0, zgodnie ze wzorem [2.15]:

```
for n in range(12):
    s=0
    for k in range(n+1):
        s=s+newton[n][k]*(-1)**k
    print(s)
```

1
0
0
0
0
0
0
0
0
0
0
0
0

A teraz liczby Fibonacciego, ukryte w trójkacie Pascala wg rys. [2.12]:

```
for n in range(12):
    s=0
    for k in range(n//2+1):
        s=s+newton[n-k][k]
    print(s)
```

```
1
1
2
3
5
8
13
21
34
55
89
144
```

1.6.2 Liczby Stirlinga I rodzaju (cyklowe)

Wzory wzięte są z tw. 4.1. Procedura jest programistycznie analogiczna do trójkąta Pascala, jedynie po lewej stronie mamy [0] i rekurencja ma inną postać:

```
stir_c=[]
stir_c.append([1])

for n in range(1,12):
    stir_c.append([0]+[stir_c[n-1][k-1]+(n-1)*stir_c[n-1][k] for
    ↪k in range(1,n)]+[1])
```

```
for n in range(9):
    row_format = "{:>6}" * (n+1)
    print(row_format.format(*stir_c[n]))
```

```
1
0      1
0      1      1
0      2      3      1
0      6      11     6      1
0      24     50     35     10     1
0      120    274    225    85     15     1
0      720    1764   1624   735    175    21     1
0      5040   13068  13132  6769   1960   322    28     1
```

Suma elementów w wierszu daje $n!$ - liczbę wszystkich permutacji:

Przykłady w języku Python

```
for n in range(0,12):
    print(sum(stir_c[n]),math.factorial(n))
```

```
1 1
1 1
2 2
6 6
24 24
120 120
720 720
5040 5040
40320 40320
362880 362880
3628800 3628800
39916800 39916800
```

Suma elementów w wierszu mnożonych naprzemiennie przez -1 daje 0, począwszy od $n = 2$

```
for n in range(12):
    s=0
    for k in range(n+1):
        s=s+stir_c[n][k]*(-1)**k
    print(s)
```

```
1
-1
0
0
0
0
0
0
0
0
0
0
```

1.6.3 Liczby Stirlinga II rodzaju (podziałowe)

Teraz bierzemy wzory z tw. 4.2:

```
stir_p=[]
stir_p.append([1])

for n in range(1,12):
    stir_p.append([0]+[stir_p[n-1][k-1]+k*stir_p[n-1][k] for k in
    range(1,n)]+[1])
```

```
for n in range(10):
    row_format = " {:>5}" * (n+1)
    print(row_format.format(*stir_p[n]))
```

1											
0	1										
0	1	1									
0	1	3	1								
0	1	7	6	1							
0	1	15	25	10	1						
0	1	31	90	65	15	1					
0	1	63	301	350	140	21	1				
0	1	127	966	1701	1050	266	28	1			
0	1	255	3025	7770	6951	2646	462	36	1		

Sumy w poszczególnych wierszach tworzą liczby Bella, zgodnie ze wzorem [4.6]:

```
for n in range(12):
    print(n, sum(stir_p[n]))
```

0	1
1	1
2	2
3	5
4	15
5	52
6	203
7	877
8	4140
9	21147
10	115975
11	678570

1.7 Zadania

Wszystkie zadania dotyczą programowania w Pythonie. Zadania „teoretyczne” znajdują się w podstawowej części książki.

1. Rozwiąż zad. [1.1] z pomocą funkcji sol_rec.
2. Utwórz funkcję realizującą rekurencję pierwszego rzędu z członem niejednorodnym postaci $a + bn$ i rozwiąż numerycznie problem dzielenia pizzy z zad. [1.2].
3. Wygeneruj liczby Fibonacciego i sprawdź numerycznie wszystkie przypadki z zad. [1.5] oraz [1.6].
4. Utwórz funkcję realizującą rekurencję z zad. [1.12].

5. Zmodyfikuj kod dla Ruiny gracza z wykładu, aby rozwiązać zad. [1.21], [1.22] i [1.23].
6. Oblicz i narysuj trójkąt liczb Bella z rys. [4.5].
7. Oblicz i narysuj trójkąt partycji $p(n, k)$ ze wzoru [4.20], rys. [4.11].
8. Oblicz i narysuj trójkąt partycji $q(n, k)$ ze wzoru [4.24], rys. [4.13].
9. Uruchom program ilustrujący algorytm Kratsuby-Ofmana (rozdz. [1.8]) ze strony www.codeandgadgets.com/karatsuba-multiplication-python/ i przedyskutuj wyniki.

ROZDZIAŁ 2

Grafy

Do analizy grafów stosujemy popularną bibliotekę `networkx`. Niniejszy kod został przygotowany z wersją 2.6.2 tego oprogramowania oraz z wersją 3.8.11 Pythona. Uwaga: wyższe wersje `networkx` niekoniecznie muszą być kompatybilne.

```
import networkx as nx # popularna biblioteka do analizy grafów
from networkx.algorithms import approximation as nalg # algorytmy na grafach
print("networkx:", nx.__version__)
!python --version
```

```
networkx: 2.6.2
Python 3.8.11
```

Pozostałe używane biblioteki są standardowe:

```
import numpy as np # numeryka

import matplotlib.pyplot as plt # grafika
import matplotlib.colors as colors
from matplotlib.patches import FancyArrowPatch, Circle
```

Ustawiamy domyślne parametry dla rozmiaru i rozdzielczości wykresów:

```
plt.rcParams["figure.figsize"] = (3,2.3) # rozmiar x i y wykresów w calach
plt.rcParams["figure.dpi"] = 120 # rozdzielcość dpi
```

2.1 Rozgrzewka

Zaczynamy od tworzenia i rysowania bardzo prostych grafów. Podstawowe definicje można znaleźć w rozdz. [5.1].

```
G = nx.Graph()    # graf pusty

G.add_node("1")  # dodanie wierzchołka "1"

# narysuj graf z parametrami grafiki
nx.draw_networkx(G, node_size=100, font_size=8, font_color='white',
                  font_weight='bold')
plt.axis('off') # usuń ramkę
plt.show()
```

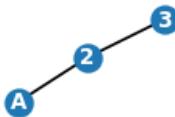
1

```
# dodaj więcej wierzchołków
G.add_node('A')          # pojedynczy wierzchołek
G.add_nodes_from([2, 3])  # zakres wierzchołków

# dodaj krawędzie
G.add_edge(2, 'A') # łączy wierzchołek 2 z 'A' itd.
G.add_edge(2, 3)

# narysuj graf z parametrami grafiki
nx.draw_networkx(G, node_size=100, font_size=8, font_color='white',
                  font_weight='bold')
plt.axis('off')
plt.show()
```

1



Rozłożenie wierzchołków

Jeśli wykonamy powyższą komórkę ponownie, układ wierzchołków może być inny. Wynika to stąd, że **networkx** ustawia je wg pewnej losowej procedury.

Dla danego grafu możemy „wyłowić” jego wierzchołki i krawędzie:

```
list(nx.nodes(G))
```

```
['1', 'A', 2, 3]
```

```
list(nx.edges(G))
```

```
[('A', 2), (2, 3)]
```

Mogemy też znaleźć stopnie wierzchołków

```
list(nx.degree(G)) # stopnie wierzchołków
```

```
[('1', 0), ('A', 1), (2, 2), (3, 1)]
```

i ich histogram:

```
nx.degree_histogram(G) # histogram stopni wierzchołków
```

```
[1, 2, 1]
```

Powyższe oznacza, że w G mamy jeden wierzchołek stopnia 0 (izolowany), dwa stopnia 1 i jeden stopnia 2.

Mogemy też jawnie zadać położenie wierzchołków na płaszczyźnie, co jest bardzo przydatne w rysowaniu grafów:

Przykłady w języku Python

```
G=nx.Graph()    # inicjalizacja - graf pusty

# zadajemy położenie i etykiety wierzchołków
G.add_node(0,pos=(0,0),label='s')
G.add_node(1,pos=(0,1),label='1')
G.add_node(2,pos=(1,0),label='2')
G.add_node(3,pos=(1,1),label='3')

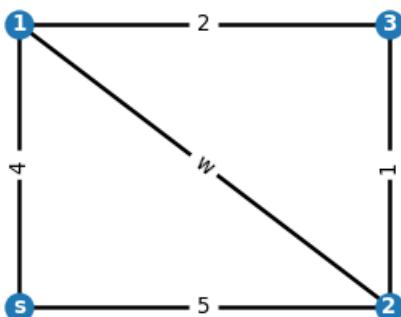
# nadajemy krawędziom wagę
G.add_edge(1,2, weight="w")
G.add_edge(0,2, weight=5)
G.add_edge(1,3, weight=2)
G.add_edge(2,3, weight=1)
G.add_edge(0,1, weight=4)

# wyciągamy atrybuty grafu
pos = nx.get_node_attributes(G,'pos')           # położenia wierzchołków
labels = nx.get_node_attributes(G,'label')        # etykiety wierzchołków
weights = nx.get_edge_attributes(G,'weight')      # wagie krawędzi

# narysuj wierzchołki
nx.draw_networkx_nodes(G, pos, node_size=100)
nx.draw_networkx_labels(G, pos, labels, font_color='white', font_weight='bold', font_size=8)

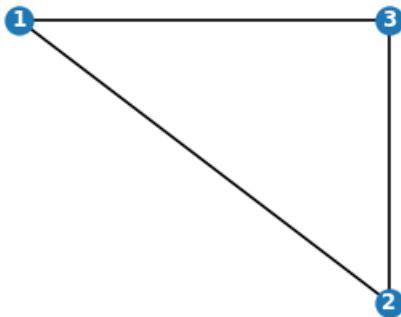
# narysuj krawędzie, tutaj etykiety utożsamiamy z wagami (edge_labels=weights)
nx.draw_networkx_edges(G, pos, width=1.5)
nx.draw_networkx_edge_labels(G, pos, edge_labels=weights, font_size=8)

plt.axis('off')                                # usuń ramkę
plt.show()
```



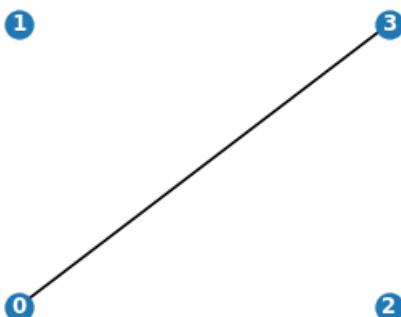
Biblioteka **networkx** posiada bardzo wiele funkcji działających na grafach. Poniżej ukazujemy kilka z nich.

```
# podgraf grafu G, zawierający wierzchołki 1,2,3
nx.draw_networkx(nx.subgraph(G, [1,2,3]), pos, labels, node_size=100,
                 font_color='white', font_weight='bold', font_
                 size=8)
plt.axis('off')
plt.show()
```



Zauważmy, że rozmieszczenie wierzchołków w powyższym podgrafie jest takie samo jak w G , dzięki użyciu atrybutu **pos**.

```
# uzupełnienie (do grafu pełnego)
nx.draw_networkx(nx.complement(G), pos, labels, node_size=100,
                 font_color='white', font_weight='bold', font_
                 size=8)
plt.axis('off')
plt.show()
```

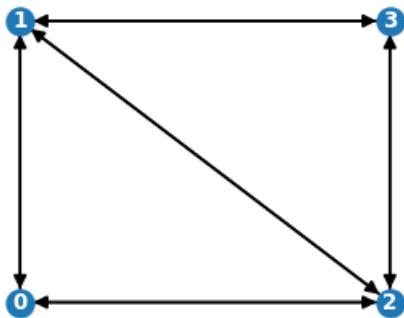


Istotnie, dodanie powyższego grafu do G uzupełnia „brakującą” krawędź i daje w wyniku graf pełny.

Przykłady w języku Python

```
# konwersja grafu nieskierowanego do skierowanego
H=nx.to_directed(G)

nx.draw_networkx(H,pos,labels,node_size=100,
                  font_color='white',font_weight='bold',font_
                  size=8)
plt.axis('off')
plt.show()
```



2.2 Mosty królewieckie

A teraz najsłynniejszy problem. Utwórzmy graf mostów królewieckich z rys. [5.5]. Ponieważ występują tu krawędzie wielokrotne, stosujemy funkcję **MultiGraph**.

```
G = nx.MultiGraph()    # multigraf dla przypadku wielokrotnych_
                        ↴krawędzi

G.add_node('A',label='A',pos=(0,-1))    # dodaj wierzchołki
G.add_node('B',label='B',pos=(0,0))
G.add_node('C',label='C',pos=(0,1))
G.add_node('D',label='D',pos=(1,0))

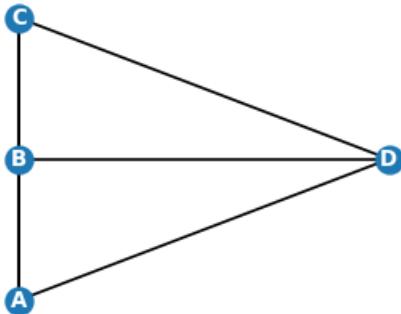
G.add_edge('A', 'B')    # dodaj krawędzie
G.add_edge('B', 'A')
G.add_edge('C', 'B')
G.add_edge('B', 'C')
G.add_edge('A', 'D')
G.add_edge('B', 'D')
G.add_edge('C', 'D')

pos=nx.get_node_attributes(G,'pos')
nx.draw_networkx(G,pos,node_size=100,
                  font_color='white',font_weight='bold',font_
                  size=8)
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
plt.axis('off')
plt.show()
```



Krawędzie wielokrotne

Powyżej niestety nie widać krawędzi wielokrotnych, które się nakładają na siebie, z czym funkcja **draw-networkx** ma kłopoty!

Rozwiązaniem jest rysowanie krawędzi wg przepisu 3 ze strony <https://www.py4u.net/discuss/22430> od użytkownika atomh33ls. Aby ten prosty przepis zrozumieć, zobaczymy najpierw, jak **MultiGraph** reprezentuje krawędzie:

```
list(G.edges)
```

```
[('A', 'B', 0),
 ('A', 'B', 1),
 ('A', 'D', 0),
 ('B', 'C', 0),
 ('B', 'C', 1),
 ('B', 'D', 0),
 ('C', 'D', 0)]
```

Każda krawędź jest krotką, np. (A, B, 0). Widzimy, że w przypadku krawędzi wielokrotnych są one rozróżniane elementem numer 2 krotki, który w naszym konkretnym przypadku przyjmuje wartości 0 lub 1. W szczególności, mamy dwie krawędzie od A do B: (A, B, 0) oraz (A, B, 1). Idea poniższego przepisu jest taka, by krzywizna krawędzi zależała od jej numeru.

```
nx.draw_networkx_nodes(G, pos, node_size = 100) # narysuje
                                                # wierzchołki
labels = nx.get_node_attributes(G, 'label')      # etykiety
                                                # wierzchołków
```

(ciąg dalszy na następnej stronie)

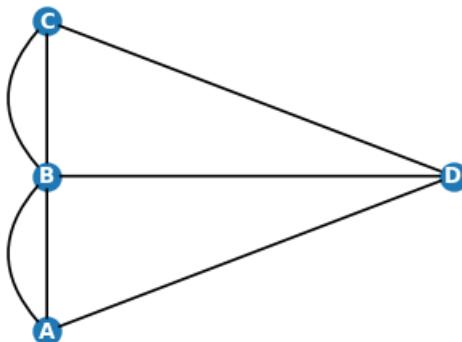
Przykłady w języku Python

(kontynuacja poprzedniej strony)

```
nx.draw_networkx_labels(G, pos, labels, font_color='white', font_
    ↪weight='bold', font_size=8)

for e in G.edges:    # pętla po krawędziach
    plt.annotate("", xy=pos[e[0]], xycoords='data',
        xytext=pos[e[1]], textcoords='data',
        arrowprops=dict(arrowstyle="-", color="black",
            shrinkA=5, shrinkB=5,
            patchA=None, patchB=None,
            # krzywizna jest proporcjonalna do e[2]
            connectionstyle="arc3,rad=rrr".replace(
    ↪'rrr', str(0.5*e[2])
        ),
        ),
    )

plt.axis('off')
plt.show()
```



Poniżej kilka definicji z rozdz. [5.1] w zastosowaniu do grafu mostów królewieckich:

```
list(nx.degree(G))  # stopnie wierzchołków
```

```
[('A', 3), ('B', 5), ('C', 3), ('D', 3)]
```

```
A=nx.adjacency_matrix(G) # macierz sąsiedztwa
print(A) # format dla tzw. macierzy rzadkiej

# nr wierzchołków liczba krawędzi
```

```
(0, 1)      2
(0, 3)      1
(1, 0)      2
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
(1, 2)      2
(1, 3)      1
(2, 1)      2
(2, 3)      1
(3, 0)      1
(3, 1)      1
(3, 2)      1
```

```
a = A.todense() # macierz sąsiedztwa w postaci zwykłej tablicy
print(a)
```

```
[[0 2 0 1]
 [2 0 2 1]
 [0 2 0 1]
 [1 1 1 0]]
```

```
list(nx.generate_adjlist(G)) # listy incydencji
```

```
['A B B D', 'B C C D', 'C D', 'D']
```

Uwaga!

Widzimy, że w **networkx** przyjęta jest bardziej oszczędna konwencja dla list incydencji niż w książce (por. **rozdz. [5.1]**). Sąsiadujące wierzchołki podawane są tylko „w jedną stronę”, np. od A do B (pierwsza lista powyżej), ale wtedy nie są już podawane od B do A (druga lista), itp. Prowadzi to do bardziej zwięzłego zapisu.

```
nx.is_eulerian(G) # graf oczywiście nie jest eulerowski
```

```
False
```

Można też utworzyć graf w oparciu o listy incydencji:

```
K=nx.MultiGraph()

K = nx.from_numpy_matrix(a, parallel_edges = True, create_using=
                           nx.MultiGraph())

# dodajemy jeszcze 2 krawędzie
K.add_edge(0,1)
K.add_edge(1,2)

print(list(nx.edges(K)))
```

```
[ (0, 1), (0, 1), (0, 1), (0, 3), (1, 2), (1, 2), (1, 2), (1, 3),  
 ↪(2, 3)]
```

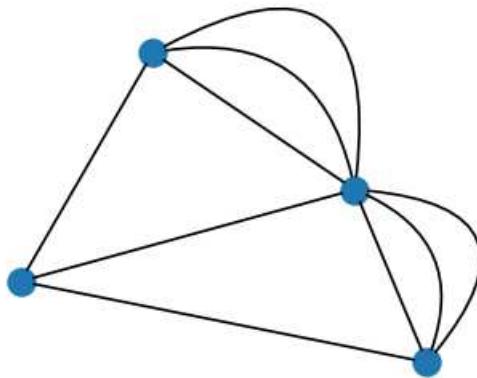
Schematy rozłożenia wierzchołków

W **networkx** możemy stosować rozmaite schematy rozłożenia wierzchołków. Dodatkowo, każdy schemat posiada element losowy, co prowadzi do innego rysunku przy kolejnych wywoływaniach funkcji rysującej graf.

Poniżej sprawdź różne schematy:

```
nx.spring_layout(G), nx.kamada_kawai_layout(G), nx.random_layout(G),  
nx.shell_layout(G), nx.circular_layout(G), nx.spectral_layout(G)
```

```
pos=nx.spring_layout(K)    # ustawienie pozycji wierzchołków w  
↪spring_layout  
nx.draw_networkx_nodes(K, pos, node_size = 100) # narysuje  
↪wierzchołki  
  
for e in K.edges:    # pętla po krawędziach, jak w poprzednim  
↪przykładzie  
    plt.annotate("",  
        xy=pos[e[0]], xycoords='data',  
        xytext=pos[e[1]], textcoords='data',  
        arrowprops=dict(arrowstyle="-", color="black",  
                        shrinkA=5, shrinkB=5,  
                        patchA=None, patchB=None,  
                        connectionstyle="arc3,rad=rrr".replace(  
↪'rrr', str(0.5*e[2]))  
        ),  
        ),  
    )  
  
plt.axis('off')  
plt.show()
```



Ponieważ stopnie dokładnie dwóch wierzchołków są nieparzyste, zgodnie z wnioskami [5.1] i [5.2] graf K nie jest eulerowski, ale jest półeulerowski:

```
list(nx.degree(K))      # stopnie wierzchołków
```

```
[ (0, 4), (1, 7), (2, 4), (3, 3) ]
```

```
nx.is_eulerian(K)      # graf K nie jest eulerowski
```

```
False
```

```
nx.is_semieulerian(K) # ... ale jest półeulerowski
```

```
True
```

A oto droga Eulera (zaczyna się w wierzchołku 3 i kończy w wierzchołku 1, gdzie obydwa są stopnia nieparzystego):

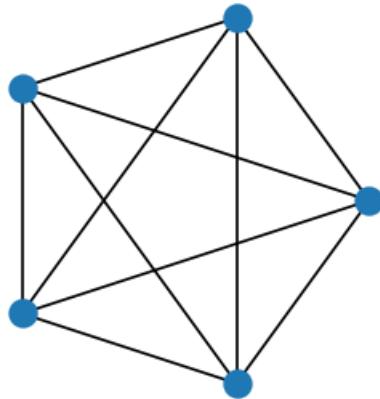
```
list(nx.eulerian_path(K))
```

```
[ (1, 0), (0, 1), (1, 0), (0, 3), (3, 1), (1, 2), (2, 1), (1, 2), ↵(2, 3) ]
```

2.3 Kalejdoskop grafów

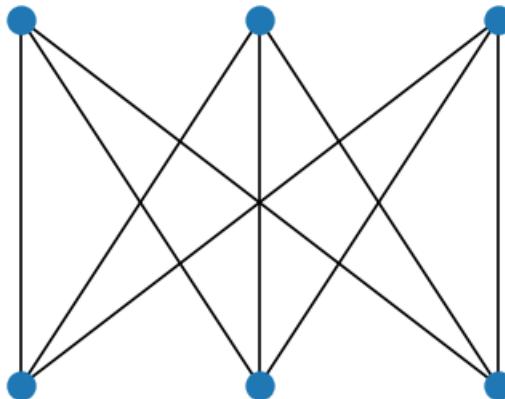
Bez trudu możemy wygenerować różne typowe grafy, zob. rozdz. [5.3]:

```
K5 = nx.complete_graph(5) # graf pełny
nx.draw_circular(K5, node_size=100)
plt.axis('equal')
plt.show()
```

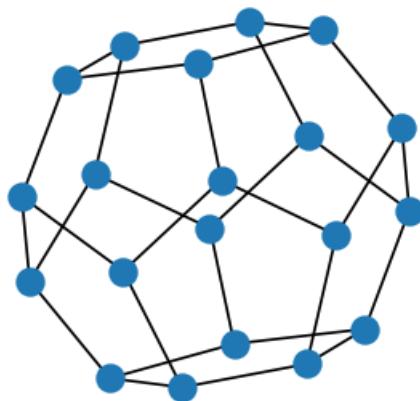


```
K33 = nx.complete_bipartite_graph(3,3) # graf dwudzielny pełny

# położenia wierzchołków zadane z pomocą słownika
posi = {0:[-1,1], 1:[0,1], 2:[1,1], 3:[-1,-1], 4:[0,-1], 5:[1,-1]}
nx.draw(K33,posi,node_size=100)
plt.show()
```



```
H = nx.dodecahedral_graph() # graf platoński (12-ścian foremny)
nx.draw(H,node_size=100)
plt.axis('equal')
plt.show()
```



Graf H jest regularny (stopień każdego wierzchołka jest taki sam i wynosi 3):

```
print(list(nx.degree(H)))
```

```
[(0, 3), (1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (6, 3), (7, 3),
 (8, 3), (9, 3), (10, 3), (11, 3), (12, 3), (13, 3), (14, 3),
 (15, 3), (16, 3), (17, 3), (18, 3), (19, 3)]
```

2.4 Badanie grafów

Poniższa funkcja podaje podstawowe informacje o grafie:

```
def graf_info(G):
    st=str(type(G))
    print("Liczba wierzchołków: ", int(G.number_of_nodes()))
    print("Liczba krawędzi: ", int(G.number_of_edges()))
    print("Lista wierzchołków: ", list(G.nodes()))
    print("Lista krawędzi: ", list(G.edges()))
    if "Di" not in st: # dla grafu
        print("Stopnie całkowite wierzchołków: ", dict(G.
        degree()))
    if "DiGraph" in st:
        print("Stopnie wejściowe wierzchołków: ", dict(G.in_
        degree()))
        print("Stopnie wyjściowe wierzchołków: ", dict(G.out_
        degree()))
```

Możemy badać różne typy grafów. Poniższą analizę należy powtórzyć dla wszystkich możliwości:

- graf nieskierowany prosty (**Graph**)

Przykłady w języku Python

- graf nieskierowany z możliwymi krawędziami wielokrotnymi i pętlami (**MultiGraph**)
- graf skierowany prosty (**DiGraph**)
- graf skierowany z możliwymi krawędziami wielokrotnymi i pętlami (**MultiDiGraph**)

```
# sprawdź po kolej i wszystkie możliwości

G=nx.Graph()
# G=nx.MultiGraph()
# G=nx.DiGraph()
# G=nx.MultiDiGraph()
```

Automatyczne dodawanie wierzchołków

Przy dodawaniu krawędzi, wierzchołki na ich końcach dodają się automatycznie, więc nie musimy wykonywać polecenia G.add_nodes(...)

```
# Tworzymy graf z wagami, nazwanymi 'dist'
G.add_weighted_edges_from([
    # dodajemy krawędź między wierzchołkami 1 i 2 o wadze 7
    (1, 2, 7),
    # dodajemy ponownie krawędź między wierzchołkami 1 i 2, tym razem o wadze 3
    # MultiGraph/MultiDiGraph ją uwzględniają, Graph/DiGraph ignorują
    # rysunek ignoruje krawędzie wielokrotne
    (1, 2, 3),
    # dodajemy krawędź między wierzchołkami 2 i 1 o wadze 1
    # MultiGraph, DiGraph, MultiDiGraph uwzględniają, Graph ignoruje
    (2, 1, 1),
    # dodajemy pętle, rysunek ignoruje pętle
    (6, 6, 3),
    # pozostałe wierzchołki
    (2, 4, 15),
    (4, 5, 6),
    (5, 6, 9),
    (6, 1, 14),
    (1, 3, 9),
    (2, 3, 10),
    (3, 4, 11),
    (3, 6, 2),
], weight='dist')

graf_info(G)
```

```
Liczba wierzchołków: 6
Liczba krawędzi: 10
Lista wierzchołków: [1, 2, 6, 4, 5, 3]
Lista krawędzi: [(1, 2), (1, 6), (1, 3), (2, 4), (2, 3), (6, 6),
    ↵ (6, 5), (6, 3), (4, 5), (4, 3)]
Stopnie całkowite wierzchołków: {1: 3, 2: 3, 6: 5, 4: 3, 5: 2, ↵
    ↵ 3: 4}
```

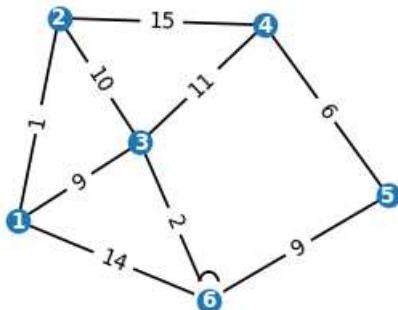
Jesteśmy gotowi narysować nasz graf!

```
nodePos = nx.spring_layout(G) # położenia wierzchołków
label_weight=nx.get_edge_attributes(G,'dist') # etykiety krawędzi

nx.draw_networkx(G, pos=nodePos, with_labels=True, font_size=8,
    ↵ font_color='white',
    ↵ font_weight='bold', node_size=70)

# rysowanie etykiet krawędzi nie działa dla MultiGraph i
# MultiDiGraph
if 'Multi' not in str(type(G)):
    nx.draw_networkx_edge_labels(G, pos=nodePos, edge_
    ↵ labels=label_weight, font_size=8);

plt.axis('off')
plt.show()
```



Mogliśmy badać wiele innych cech grafu, np.

```
# sąsiedzi wskazanego wierzchołka
print("Lista wierzchołków osiągalnych z 1 w pojedynczym kroku:", ↵
    ↵ list(G.neighbors(1)))
```

```
Lista wierzchołków osiągalnych z 1 w pojedynczym kroku: [2, 6, 3]
```

```
nx.is_biconnected(G)    # dwuspójność, tj. trzeba przeciąć min 2-  
↪krawędzie, by rozspójnić
```

```
True
```

2.5 Algorytmy na grafach

2.5.1 Przeszukiwanie wszerz

W rozdz. [5.7] i [5.8] omówiono niektóre podstawowe algorytmy stosowane w analizie grafów. Poniżej podajemy „autorski” program dla algorytmu [5.3] przeszukiwania grafu wszerz. Algorytm sprawdza też „przy okazji”, czy graf jest spójny i w tym przypadku zwraca drzewo spinające grafu. Jeśli graf nie jest spójny, zwracane jest drzewo spinające części spójnej zawierającej wierzchołek startowy przeszukiwania.

```
def wszerz(GG, deb=False) :  
    """  
    przeszukiwanie wszerz  
  
    input:  
        GG - graf w formacie Graph  
        deb - boolean, kontrola wydruku  
  
    output:  
        jeśli GG spójny - drzewo spinające  
        jeśli GG niespójny - drzewo spinające części spójnej  
        ↪zawierającej wierzchołek startowy  
    """  
  
    G=GG.copy()          # kopia grafu (będzie "psuta")  
    T=nx.Graph()         # zainicjalizuj puste drzewo spinające  
    V=list(nx.nodes(G)) # lista wierzchołków nieodwiedzonych  
    V2=[]               # pusta lista wierzchołków odwiedzonych  
  
    V2.append(V.pop(0)) # przenieś wierzchołek 0 z  
    ↪nieodwiedzonych do odwiedzonych  
  
    if len(V) == 0:      # specjalny przypadek, gdy G ma tylko  
    ↪jeden wierzchołek  
        if deb:  
            print('jeden wierzchołek')  
        T.add_nodes_from(V2)  
        return T  
  
    lv=len(V) # obecna liczba wierzchołków nieodwiedzonych  
    lv2=lv+1  # cokolwiek innego od lv, by poniższa pętla ruszyła
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```

while len(V)>0 and lv != lv2: # wykonuj, dopóki są
    ↵wierzchołki nieodwiedzone
                                # i ich liczba się zmienia, tj.
    ↵ lv != lv2

    lv=lv2                  # obecna liczba wierzchołków lv
    ↵nieodwiedzonych = poprzedniej
    VV2=V2.copy()           # skopiuj listę nieodwiedzonych

    for v2 in VV2:          # pętla po wierzchołkach odwiedzonych
        VV=V.copy()           # skopiuj listę nieodwiedzonych

    for v in VV:            # pętla po wierzchołkach
    ↵nieodwiedzonych
        if (v2, v) in G.edges(): # jeśli odwiedzony i
    ↵nieodwiedzony sąsiadują
            V.remove(v)      # przenieś wierzchołek z
    ↵nieodwiedzonych
            V2.append(v)     # do odwiedzonych
            G.remove_edge(v2,v) # usuń krawędź z grafu
            T.add_edge(v2,v)   # i dodaj ją do drzewa
            lv2=lv-1           # liczba
    ↵nieodwiedzonych zmniejsza się o 1

# To się wykona, jeśli graf jest niespójny. Wówczas lv się nie
    ↵zmienia, a istnieją nieodwiedzone!
    if len(V)>0:
        if deb:
            print('niespójny!')

return T

```

Test algorytmu na przykładowym grafie:

```

G = nx.Graph()

# G.add_nodes_from([0])

G.add_nodes_from([0,1,2,3,4,5,6,7,8,9, 'A', 'B', 'C', 'r'])
G.add_edges_from([(0,0), (1,3), (3,1), (5,0), (5, 'A'),
    (2,3), (4,3), (2,4), (7,4), (7,4), (6,9), (5,8), (0,
    ↵1), (0,9), (0,9), (1,9), ('B', 'C'), (7,8)])

```

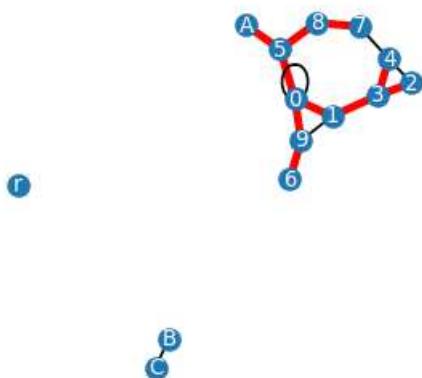
H=G.copy()
T=wszerz(H, **True**)

niespójny!

Przykłady w języku Python

```
plt.figure(figsize=(3,3))
nodePos = nx.spring_layout(H)
nx.draw_networkx(H, pos=nodePos, with_labels=True,
                 node_size=60, font_color='white', font_size=8)
nx.draw_networkx_edges(T, pos=nodePos, edge_color='r', width=3)

plt.axis('equal')
plt.axis('off')
plt.show()
```



Widzimy uzyskane drzewo spinające, oznaczone na czerwono, dla składowej spójnej grafu zawierającej wierzchołek startowy 0.

Możemy teraz łatwo utworzyć funkcję sprawdzającą ile spójnych składowych ma graf.

```
def ile_skl(GG):
    G=GG.copy() # skopiuj graf, bo będzie się zmieniać
    i=0          # początkowa liczba składowych grafu

    while nx.number_of_nodes(G)>0: # wykonuj, dopóki G jest niepusty
        T=wszerz(G)
        G.remove_nodes_from(T) # usuń uzyskane drzewo z grafu
        i=i+1                  # liczba składowych wzrasta o 1

    print('liczba składowych spójnych:',i)
    return i                  # ostateczna liczba składowych
```

```
ile_skl(G)
```

```
liczba składowych spójnych: 3
```

3

Podobnie, możemy uzyskać **las drzew spinających**, sumując drzewa spinające poszczególnych składowych spójnych grafu:

```
def las(GG):
    G=GG.copy()    # skopiuj graf, bo będzie się zmieniać
    L=nx.Graph()  # las
    i=0            # początkowa liczba składowych grafu

    while nx.number_of_nodes(G)>0: # wykonuj, dopóki G jest
        ↪niepusty
            T=wszerz(G)
            G.remove_nodes_from(T)      # usuń uzyskane drzewo z
        ↪grafu
            L=nx.union(L,T)          # dodaj drzewo T do lasu L
            i=i+1                    # liczba składowych wzrasta o
        ↪1
            print('liczba składowych spójnych:',i)
    return L      # las spinający
```

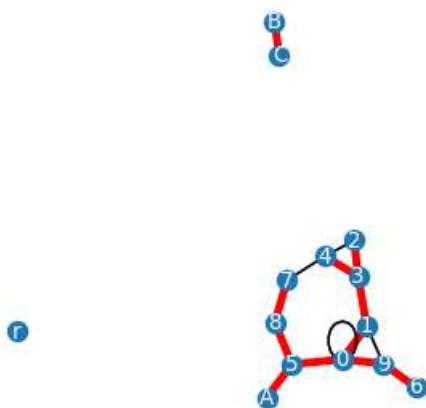
```
lasek=las(G)
graf_info(lasek)
```

```
liczba składowych spójnych: 3
Liczba wierzchołków: 14
Liczba krawędzi: 11
Lista wierzchołków: [0, 1, 5, 9, 3, 8, 'A', 6, 2, 4, 7, 'B', 'C',
↪, 'r']
Lista krawędzi: [(0, 1), (0, 5), (0, 9), (1, 3), (5, 8), (5, 'A
↪'), (9, 6), (3, 2), (3, 4), (8, 7), ('B', 'C')]
Stopnie całkowite wierzchołków: {0: 3, 1: 2, 5: 3, 9: 2, 3: 3,
↪8: 2, 'A': 1, 6: 1, 2: 1, 4: 1, 7: 1, 'B': 1, 'C': 1, 'r': 0}
```

```
plt.figure(figsize=(3, 3))

nodePos = nx.spring_layout(G)
nx.draw_networkx(G, pos=nodePos, with_labels=True,
                  node_size=50, font_color='white', font_size=8)
nx.draw_networkx_edges(lasek, pos=nodePos, edge_color='r', width=3)

plt.axis('equal')
plt.axis('off')
plt.show()
```



2.5.2 Algorytmy wbudowane w networkx

Algorytmy networkx

Biblioteka **networkx** zawiera wiele bardzo użytecznych algorytmów działających na grafach. Są one „profesjonalne” i należy ich używać, gdy tylko możliwe. Powyższy algorytm „autorski” miał na celu pokazanie, że algorytmy na grafach i implementujące je programy są w swojej istocie proste i łatwe w zrozumieniu.

Utwórzmy przykładowy graf skierowany i przeszukajmy go w głąb i wszerz:

```
G = nx.DiGraph()

G.add_nodes_from([0, 1, 2, 3, 4, 5, 6])
G.add_edges_from([
    (0, 1),
    (0, 2),
    (0, 4),
    (1, 3),
    (2, 3),
    (3, 4),
    (3, 5),
    (4, 6),
    (5, 6),
])

nodePos = nx.circular_layout(G)

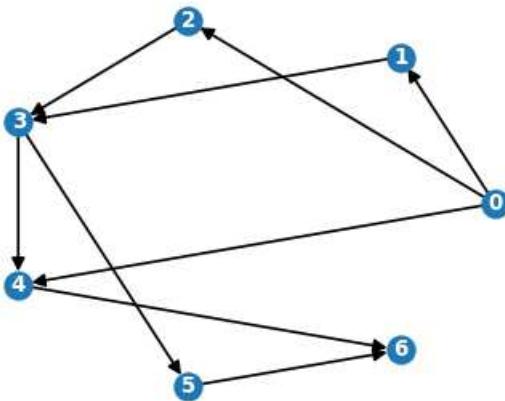
nx.draw(G, with_labels=True, pos=nodePos, font_color='white', 
    font_weight='bold',
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

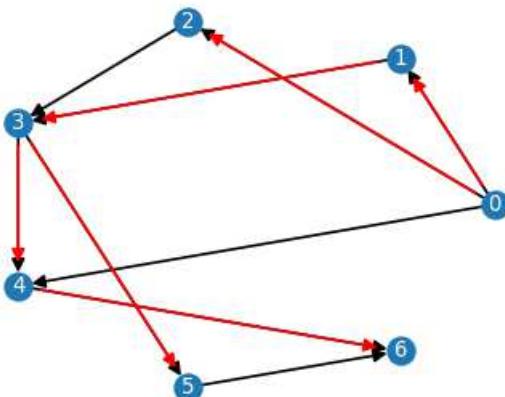
```
node_size=100, font_size=8)
```

```
plt.show()
```



```
# przeszukiwanie w głąb
T = nx.depth_first_search.dfs_tree(G, 0)

nx.draw(G, with_labels=True, pos=nodePos, node_size=100, font_color=
    'white', font_size=8)
nx.draw_networkx_edges(T, pos=nodePos, edge_color='r')
plt.show()
```

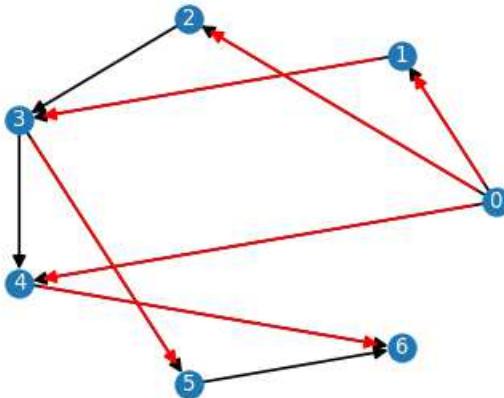


```
# przeszukiwanie wszerz
T = nx.breadth_first_search.bfs_tree(G, 0)
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
nx.draw(G, with_labels=True, pos=nodePos, node_size=100, font_color='white', font_size=8)
nx.draw_networkx_edges(T, pos=nodePos, edge_color='r')
plt.show()
```



Możemy łatwo sprawdzić na powyższych rysunkach realizację alg. [5.2] i [5.3].

Kolejne algorytmy dotyczą grafów z wagami, **rozdz. [5.8]**. Tworzymy zatem przykładowy graf z wagami i znajdujemy najkrótszą ścieżkę między dwoma wierzchołkami:

```
G = nx.Graph()
G.add_nodes_from([1, 2, 3, 4, 5, 6])
G.add_weighted_edges_from([
    (1, 2, 7),
    (2, 4, 15),
    (4, 5, 6),
    (5, 6, 9),
    (6, 1, 14),
    (1, 3, 9),
    (2, 3, 10),
    (3, 4, 11),
    (3, 6, 2),
], weight='dist')

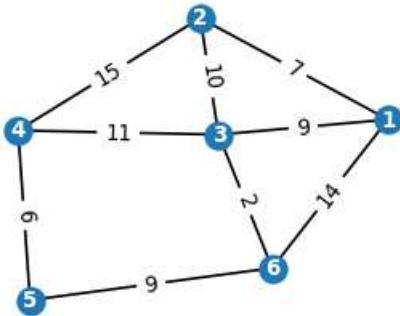
nodePos = nx.spring_layout(G)
label_weight=nx.get_edge_attributes(G, 'dist')

nx.draw_networkx(G, pos=nodePos, with_labels=True,
                 node_size=100, font_color='white', font_size=8,
                 font_weight='bold')
nx.draw_networkx_edge_labels(G, pos=nodePos, edge_labels=label_
                             weight, font_size=8)
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
plt.axis("off")
plt.show()
```



```
# najkrótsza ścieżka między wierzchołkami 1 i 5
P = nx.shortest_path(G, 1, 5, weight='dist')
print(P)
```

```
[1, 3, 6, 5]
```

```
# sumaryczna waga (długość) najkrótszej ścieżki
print(nx.shortest_path_length(G, 1, 5, weight='dist'))
```

```
20
```

```
# najkrótszą ścieżkę oznaczymy na czerwono
red_edges = list(zip(P,P[1:])) # zip stwarza odpowiednią krotkę
# z listy P
print(red_edges)
```

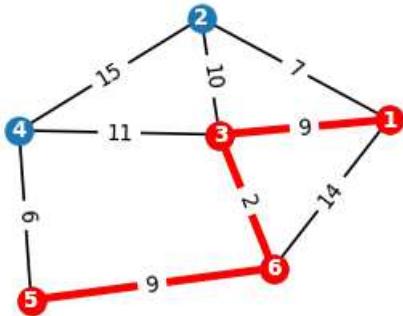
```
[(1, 3), (3, 6), (6, 5)]
```

```
sh=nx.Graph()          # graf dla najkrótszej ścieżki
sh.add_nodes_from(P)
sh.add_edges_from(red_edges)
nx.draw_networkx(G, pos=nodePos, with_labels=True,
                  node_size=100, font_color='white', font_size=8,
                  font_weight='bold')
nx.draw_networkx_edge_labels(G, pos=nodePos, edge_labels=label_
weight, font_size=8)
nx.draw_networkx(sh, pos=nodePos, node_color='r',
                  node_size=100, font_color='white', font_size=8,
                  font_weight='bold')
nx.draw_networkx_edges(sh, pos=nodePos, edge_color='r', width=3)
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
plt.axis("off")
plt.show()
```



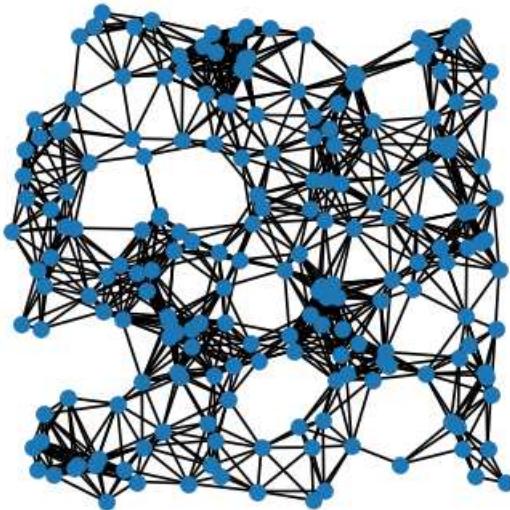
Grafy napotykane w „codziennym życiu” są na ogół bardzo duże (zob. wstęp **rozdz. [5]**). Poniżej przykład działania algorytmu znajdowania najkrótszej ścieżki dla dużego geometrycznego grafu losowego.

Geometryczny graf losowy

W konstrukcji geometrycznego grafu losowego umieszcza się jednorodnie losowo n wierzchołków w kwadracie $[0, 1] \times [0, 1]$. Dwa wierzchołki są łączone krawędzią, jeśli ich odległość jest mniejsza niż pewna zadana odległość d .

```
# geometryczny graf losowy, n=200, d=0.15
G = nx.random_geometric_graph(200, 0.15)
pos = nx.get_node_attributes(G, 'pos')

plt.figure(figsize=(4,4))
nx.draw_networkx(G, pos=pos, with_labels=False, node_size=30)
plt.axis('off')
plt.show()
```

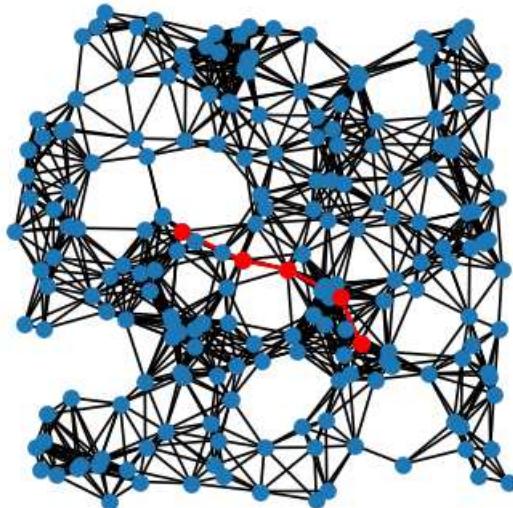


```
# najkrótsza ścieżka między wierzchołkami 1 i 50
P = nx.shortest_path(G, 10, 190)

red_edges = list(zip(P,P[1:]))

plt.figure(figsize=(4,4))
sh=nx.Graph()
sh.add_nodes_from(P)
sh.add_edges_from(red_edges)
nx.draw_networkx(G, pos=pos, with_labels=False, node_size=30)
nx.draw_networkx(sh, pos=pos, node_color='r', node_size=30, with_
    _labels=False)
nx.draw_networkx_edges(sh, pos=pos, edge_color='r', width=1.5)

plt.axis('off')
plt.show()
```



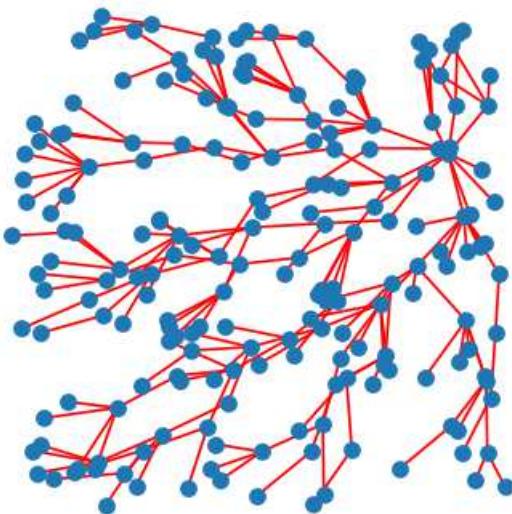
```
ile_skl(G)
```

```
liczba składowych spójnych: 1
```

```
1
```

```
# drzewo spinające
T=wszerz(G)

plt.figure(figsize=(4,4))
nx.draw_networkx(G, pos=pos, with_labels=False, node_size=30)
nx.draw_networkx(T, pos=pos, with_labels=False, node_size=30, edge_color='r')
plt.axis('off')
plt.show()
```



Średnica grafu

Średnica grafu to maksimum po wszystkich parach wierzchołków z ich najkrótszej odległości.

```
# średnica grafu (działa tylko dla grafu spójnego)
if ile_skl(G)==1:
    print("średnica grafu:", nx.diameter(G))
```

```
liczba składowych spójnych: 1
średnica grafu: 11
```

Klika

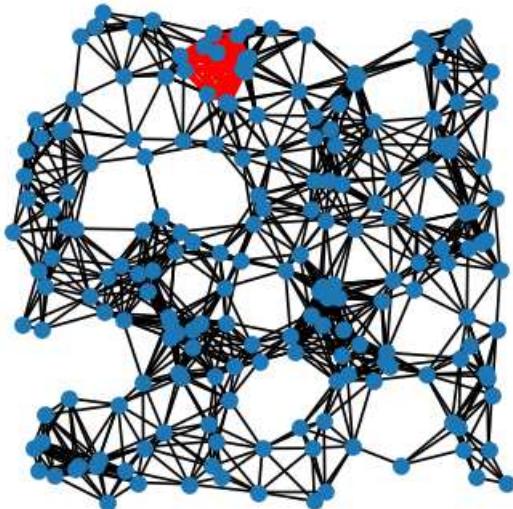
Klika to podgraf pełny - każdy zna każdego!

```
# największa klika
kli=list(nx.enumerate_all_cliques(G))[-1] # ostatni element
# listy klik
len(kli)
```

```
13
```

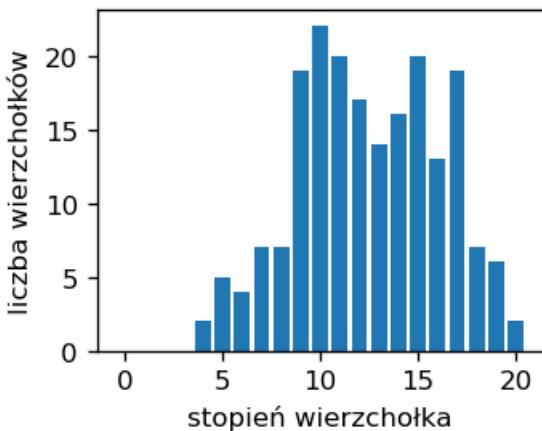
Przykłady w języku Python

```
plt.figure(figsize=(4,4))
nx.draw_networkx(G, pos, with_labels=False, node_size=30)
nx.draw_networkx(G.subgraph(kli), pos, with_labels=False, node_
    size=8,
                  edge_color='r', width=1.5)
plt.axis("off")
plt.show()
```



```
kG=nx.degree_histogram(G) # histogram stopni wierzchołków
plt.xlabel('stopień wierzchołka')
plt.ylabel('liczba wierzchołków')

plt.bar(range(len(kG)), kG)
plt.show()
```



2.6 Grafy z zewnętrznych danych

Często kroć pragniemy zanalizować dane z pomocą grafów. Dane są zebrane w plikach, więc pierwszym zadaniem jest wczytać i przekształcić je w odpowiedni format grafu. Poniżej przedstawiamy dwa popularne przykłady.

2.6.1 Sioux Falls

Ten przykład zaczerpnięty jest ze strony <https://github.com/bstabler/TransportationNetworks/tree/master/SiouxFalls> i dotyczy analizy transportu w amerykańskim mieście Sioux Falls. Plik z danymi dot. położenia wierzchołków (skrzyżowań ulic) ma format

Node X Y ;

1 50000 510000 ;

2 320000 510000 ;

3 50000 440000 ;

4 130000 440000 ;

....,

gdzie X i Y są geograficznymi współrzednymi. Wczytanie wierzchołków odbywa się następująco:

```
S=nx.Graph() # inicjalizacja grafu
```

Przykłady w języku Python

```
f = open("SiouxFalls/SiouxFalls_node.tntp", "r")
line = f.readline()          # opuść pierwszą linię
line = f.readline()
while len(line)-1:           # jeśli linia nie jest pusta
    line = line.strip(';')   # usuń znak;
    l = line.split()         # utwórz z linii tablicę
    node = int(l[0])          # nr wierzchołka
    pos1 = float(l[1])/10000  # współrzędna x przeskalowana
    pos2 = float(l[2])/10000  # współrzędna y przeskalowana
    S.add_node(node, pos=(pos1,pos2)) # dodaj wierzchołek do grafu
    line = f.readline()       # przeczytaj nową linię
f.close()

print("liczba wierzchołków:", S.number_of_nodes())
```

```
liczba wierzchołków: 24
```

Format pliku z krawędziami ma postać

From To Volume Capacity Cost

```
1 2 4494.6576464564205 6.0008162373543197
```

```
1 3 8119.079948047809 4.0086907502079407
```

```
2 1 4519.079948047809 6.0008341229953821
```

```
2 6 5967.3363961713767 6.5735982553868011
```

...,

gdzie pierwsze dwie kolumny podają wskaźniki wierzchołków, trzecia (Volume Capacity) to przepustowość komunikacyjna (w setkach samochodów na dobę) - tej wielkości nie używamy, a czwarta to koszt przejazdu.

```
f = open("SiouxFalls/SiouxFalls_flow.tntp", "r")
line = f.readline()
line = f.readline()
while len(line)-1:
    l = line.split()
    fromnode = int(l[0])
    to = int(l[1])
    volume = float(l[2])
    cost = int(float(l[3]))
    S.add_edge(fromnode, to, weight = cost) # dodaj krawędź do grafu, waga to koszt
    line = f.readline()
f.close()

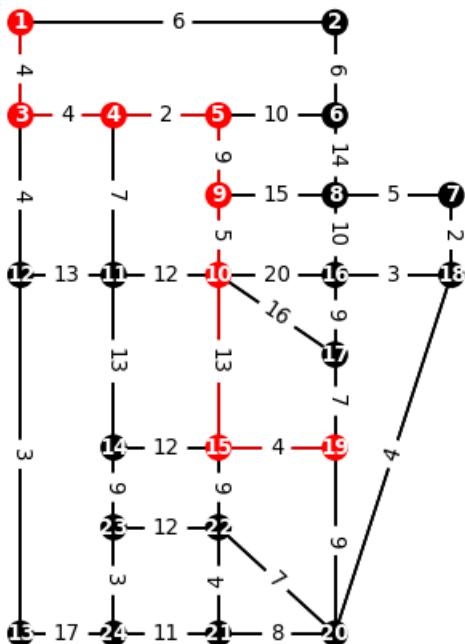
print("liczba krawędzi:", S.number_of_edges())
```

```
liczba krawędzi: 38
```

Mając zbudowany graf w oparciu o dane, możemy analizować go przy pomocy posiadanych narzędzi, w szczególności możemy z pomocą wbudowanego algorytmu Dijkstry (zob. alg. [5.6]) znaleźć najkrótszą drogę między dwoma wybranymi wierzchołkami:

```
node_pos=nx.get_node_attributes(S,'pos')           # położenia
˓→wierzchołków
arc_weight=nx.get_edge_attributes(S,'weight')      # wagi krawędzi
sp = nx.dijkstra_path(S,source = 1, target = 19) # najkrótsza
˓→droga od 1 do 19
red_edges = list(zip(sp,sp[1:]))                  # lista
˓→czerwonych krawędzi
node_col = ['black' if not node in sp else 'red' for node in S.
˓→nodes()]
˓→ # czerwony kolor dla wierzchołków na najkrótszej
˓→ścieżce, czarny poza
edge_col = ['black' if not edge in red_edges else 'red' for edge
˓→in S.edges()]
˓→ # czerwony kolor dla krawędzi na najkrótszej ścieżce,
˓→ czarny poza

# rysunek, jak w uprzednich przypadkach
plt.figure(figsize=(3.5,5))
nx.draw_networkx(S, node_pos,node_color= node_col,
                  node_size=80,font_size=8,font_color="white",
˓→font_weight='bold')
nx.draw_networkx_edges(S, node_pos,edge_color= edge_col)
nx.draw_networkx_edge_labels(S, node_pos, edge_labels=arc_weight,
˓→font_size=8)
plt.axis('off')
plt.show()
```



```
# najkrótsza ścieżka od 1 do 19  
sp
```

```
[1, 3, 4, 5, 9, 10, 15, 19]
```

```
print("koszt całkowity wzdłóż optymalnej ścieżki:",  
      nx.dijkstra_path_length(S, 1, 19, 'weight'))
```

```
koszt całkowity wzdłóż optymalnej ścieżki: 41
```

2.6.2 Sieci społeczne

Drugi przykład pochodzi ze strony <https://www.datacamp.com/community/tutorials/social-network-analysis-python> i dotyczy sieci społecznej **facebook**. Wierzchołkami są osoby, które są połączone krawędzią jeśli są znajomymi. Plik z danymi ma bardzo prosty format, gdzie każda linia to para znajomych:

...

16 329

16 331

16 332

17 19

...

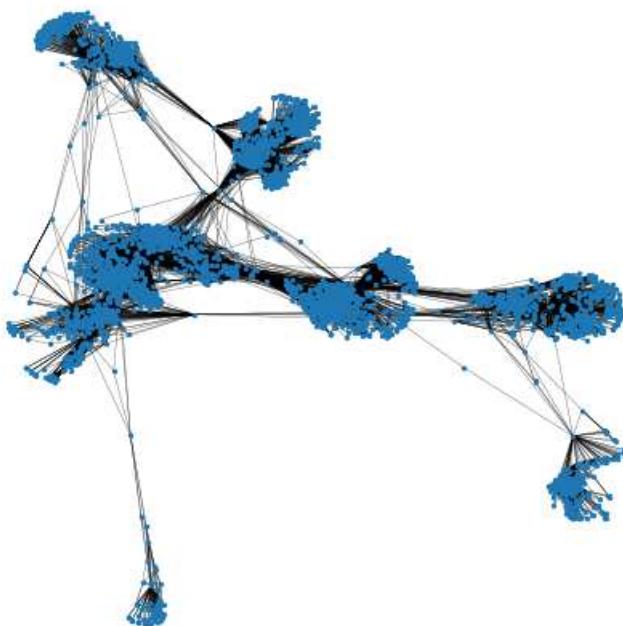
Osoba 16 zna osobe 329 itd. W tym przypadku możemy bardzo łatwo utworzyć graf poprzez instrukcję

```
G_fb=nx.read_edgelist("fb/facebook_combined.txt",
                      create_using = nx.Graph(), nodetype=int)
```

```
print(nx.info(G_fb)) # zwięzła informacja o grafie
```

```
Graph with 4039 nodes and 88234 edges
```

```
plt.figure(figsize=(5,5))
nx.draw_networkx(G_fb, with_labels=False, node_size=1, width=0.15)
plt.axis('off')
plt.show()
```

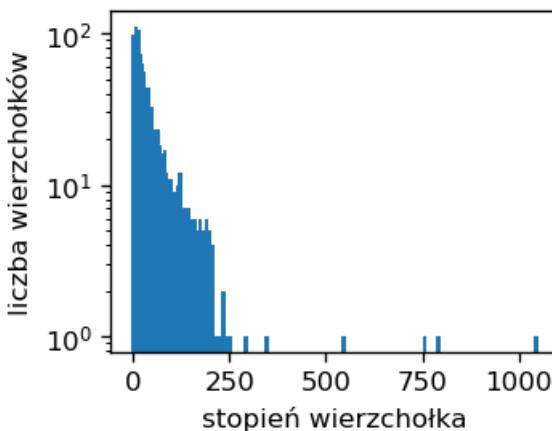


Zwróćmy uwagę na kilka wierzchołków o stosunkowo bardzo wysokim stopniu. W teorii

Przykłady w języku Python

sieci społecznych takie wierzchołki nazywamy „gwaizdowami” (stars).

```
Gh=nx.degree_histogram(G_fb)
plt.xlabel('stopień wierzchołka')
plt.ylabel('liczba wierzchołków')
plt.bar(range(len(Gh)), Gh, log=True, width=10)
plt.show()
```



2.7 Kolorowanie grafów

Kolorowanie wierzchołkowe grafów, opisane w rozdz. [5.12], uzyskujemy dzięki funkcji **equitable_color** z biblioteki **networkx**:

```
G=nx.dodecahedral_graph()                      # przykładowy graf
kolo=nx.equitable_color(G, num_colors=4)        # kolorowanie
                                                # wierzchołków
kolo_list=list(kolo.values())
print(kolo)
print("")
print(kolo_list)
```

```
{0: 0, 1: 3, 2: 0, 3: 1, 4: 2, 5: 1, 6: 2, 7: 1, 8: 0, 9: 3, 10: 2,
 11: 3, 12: 2, 13: 1, 14: 0, 15: 3, 16: 0, 17: 1, 18: 2, 19: 3}
```

```
[0, 3, 0, 1, 2, 1, 2, 1, 0, 3, 2, 3, 2, 1, 0, 3, 0, 1, 2, 3]
```

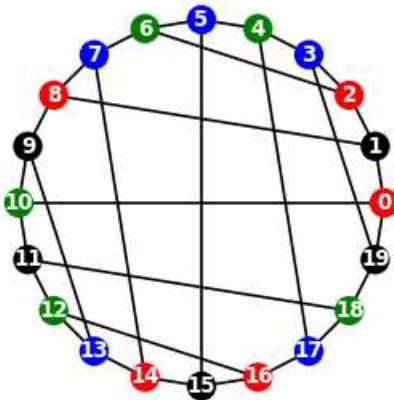
Wynikiem algorytmu jest słownik kolorów, numerowanych 0, 1, 2, ..., który skonwertowaliśmy na listę. Wskaźnikom 0, 1, 2, ... przydzielimy ulubione kolory:

```
colors=['red','blue','green','black','gray','brown','pink',
↪'yellow','orange','magenta']
```

Możemy teraz ładnie narysować pokolorowany graf:

```
cmap=[colors[kolo_list[i]] for i in range(len(kolo_list))]
nx.draw_circular(G, node_color = cmap, node_size=100, with_
↪labels=True,
            font_color='white', font_weight='bold', font_
↪size=8)

plt.axis('equal') # zachowaj proporcje osi x i y
plt.axis('off')
plt.show()
```



2.8 Przepływ w sieci

Do znajdowania maksymalnego przepływu w sieci o zadanych przepustowościach (zob. rozdz. [5.19]) służy funkcja **maximum_flow** z biblioteki **networkx**. Utwórzmy przykładową sieć:

```
G = nx.DiGraph()

G.add_edge('Z','a', capacity=3)
G.add_edge('Z','b', capacity=1)
G.add_edge('Z','d', capacity=1)
G.add_edge('a','c', capacity=3)
G.add_edge('b','c', capacity=5)
G.add_edge('b','d', capacity=4)
G.add_edge('d','e', capacity=2)
```

(ciąg dalszy na następnej stronie)

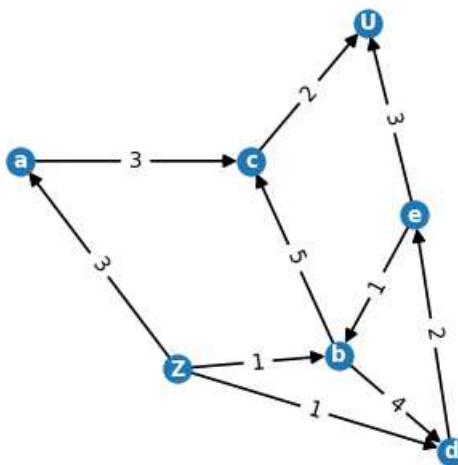
Przykłady w języku Python

(kontynuacja poprzedniej strony)

```
G.add_edge('c', 'U', capacity=2)
G.add_edge('e', 'U', capacity=3)
G.add_edge('e', 'b', capacity=1)

label_cap=nx.get_edge_attributes(G, 'capacity') # etykieta
# krawędzi = przepustowość
nodePos = nx.spring_layout(G)

plt.figure(figsize=(3.5,3.5))
nx.draw_networkx(G, pos=nodePos, node_size=100, with_labels=True,
                  font_color='white',
                  font_weight='bold', font_size=8)
nx.draw_networkx_edge_labels(G, pos=nodePos, edge_labels=label_
                             _cap, font_size=8);
plt.axis('off')
plt.show()
```



Znajdujemy teraz maksymalny przepływ między źródłem Z i ujściem U :

```
flow_value, flow_dict = nx.maximum_flow(G, 'Z', 'U') #
# maksymalny przepływ
print("maksymalny przepływ:", flow_value)
print("")
print(flow_dict)
```

maksymalny przepływ: 4

```
{'Z': {'a': 2, 'b': 1, 'd': 1}, 'a': {'c': 2}, 'b': {'c': 0, 'd':
#': 1}, 'd': {'e': 2}, 'c': {'U': 2}, 'e': {'U': 2, 'b': 0}, 'U
#: {}}
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

Należy teraz przekształcić powyższy słownik na graf, co czynimy w następujący sposób:

```
tab=[]
F=nx.DiGraph()
for v in flow_dict:
    for w in flow_dict.get(v):
        tab.append((v,w,flow_dict.get(v).get(w)))

F.add_weighted_edges_from(tab)
```

Następnie rysujemy sieć z maksymalnym przepływem. Czarne etykiety oznaczają przepustowość krawędzi, a czerwone maksymalny przepływ.

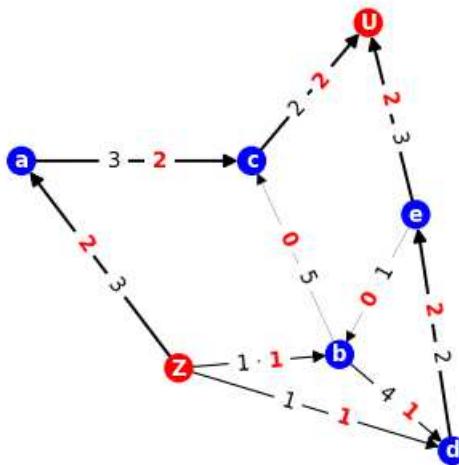
```
label_fl=nx.get_edge_attributes(F,'weight')      # etykiety_
# maksymalnego przepływu

wid=[(1.5*i+0.4)/3 for i in list(nx.get_edge_attributes(F,'weight'
            .values()))]                         # szerokość linii_
# proporcjonalna do przepływu
# kolory dla wierzchołków
color_map = []
for node in G:
    if node == 'Z' or node == 'U':
        color_map.append('red')
    else:
        color_map.append('blue')

plt.figure(figsize=(3.5,3.5))
nx.draw_networkx(G, pos=nodePos, node_color = color_map, node_
    _size=100,
    with_labels=True, font_color='white', font_weight='bold',
    font_size=8, width=wid)
nx.draw_networkx_edge_labels(G, pos=nodePos, edge_labels=label_-
    _cap, label_pos=0.6,
                           font_size=8);
nx.draw_networkx_edge_labels(F, pos=nodePos, edge_labels=label_-
    _fl,label_pos=0.4,
                           font_color='red', font_weight='bold
                           ',font_size=8);

plt.title('Przepływ maksymalny',size=10)
plt.axis('off')
plt.show()
```

Przepływ maksymalny



Znajdziemy teraz minimalny przekrój (por. tw. [5.31]):

```

mc=nx.minimum_cut(G, 'Z','U', capacity='capacity')
print(mc)
minp=mc[0]
  
```

```
(4, ({'c', 'a', 'b', 'd', 'z'}, {'e', 'U'}))
```

```

color_map = [] # kolory dla wierzchołków
for node in G:
    if node in mc[1][0]:
        color_map.append('green')
    else:
        color_map.append('gray')

nx.draw_networkx(G, pos=nodePos, node_color = color_map, node_
    _size=100, with_labels=True,
                font_color='white', font_weight='bold', font_
    _size=8, width=wid)
nx.draw_networkx_edge_labels(G, pos=nodePos, edge_labels=label_
    _cap,
                label_pos=0.6, font_size=8);
nx.draw_networkx_edge_labels(F, pos=nodePos, edge_labels=label_
    _fl,
                label_pos=0.4, font_color='red', font_size=8, font_
    _weight='bold');

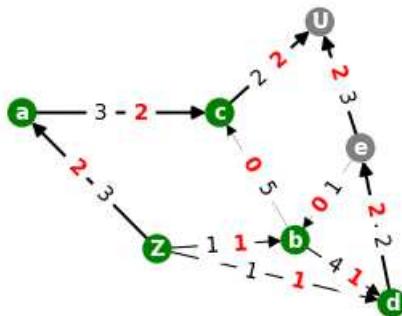
plt.title('Minimalny przekrój = '+str(mc[0]), size=10)
plt.axis('off')
  
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

plt.show()

Minimalny przekrój = 4



Wszystko się zgadza: maksymalny przepływ równa się minimalnemu przekrojowi!

2.9 Drzewo Steinera

Uwaga!

Jest to inna koncepcja drzewa Steinera niż w książce. Nie szukamy tu położenia dodatkowych wierzchołków Steinera.

Problem drzewa Steinera

W spójnym grafie G z ustalonymi wagami szukamy minimalnego drzewa S zawierającego wszystkie zadane wierzchołki „terminalowe” T należące do G. “

```
# przykładowy graf z wagami
G=nx.Graph()

G.add_weighted_edges_from([
    (0, 1, 1),
    (0, 2, 3),
    (0, 3, 1),
    (1, 2, 1),
    (1, 3, 1),
    (2, 3, 1),
    (4, 2, 0.5),
    (4, 1, 0.3),
```

(ciąg dalszy na następnej stronie)

Przykłady w języku Python

(kontynuacja poprzedniej strony)

```
(4, 0, 0.6)
])

# atrybuty
pos = nx.spring_layout(G)
weights = nx.get_edge_attributes(G, 'weight')
labels = nx.get_node_attributes(G, 'label')

T=[3,2,1] # wierzchołki "terminalowe" (drzewo S z definicji musi
    # je zawierać)

# drzewo Steinera (uwaga: działa tylko dla grafów
    # nieskierowanych)
S=nalg.steiner_tree(G, T ,weight='weight')

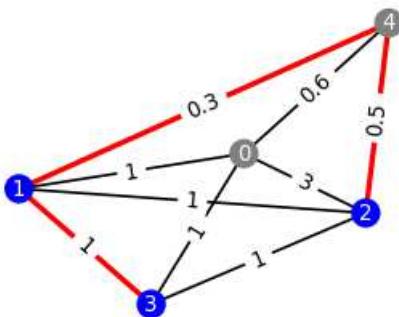
# kolory dla wierzchołków terminalowych
color_map = []
for node in G:
    if node in T:
        color_map.append('blue')
    else:
        color_map.append('gray')

nx.draw_networkx(G,pos,with_labels=True,node_size=100,font_color=
    'white',
                  node_color = color_map,font_size=8)
nx.draw_networkx_edge_labels(G,pos,edge_labels=weights,font_
    size=8);
nx.draw_networkx_edges(S,pos,edge_color='r',width=2)

le=0
for e in S.edges:
    le+=S.get_edge_data(*e) ['weight'] # * zwraca tzw. "tuple form
    "
print("waga drzewa Steinera:", le)

plt.axis('off')
plt.show()
```

```
waga drzewa Steinera: 1.8
```

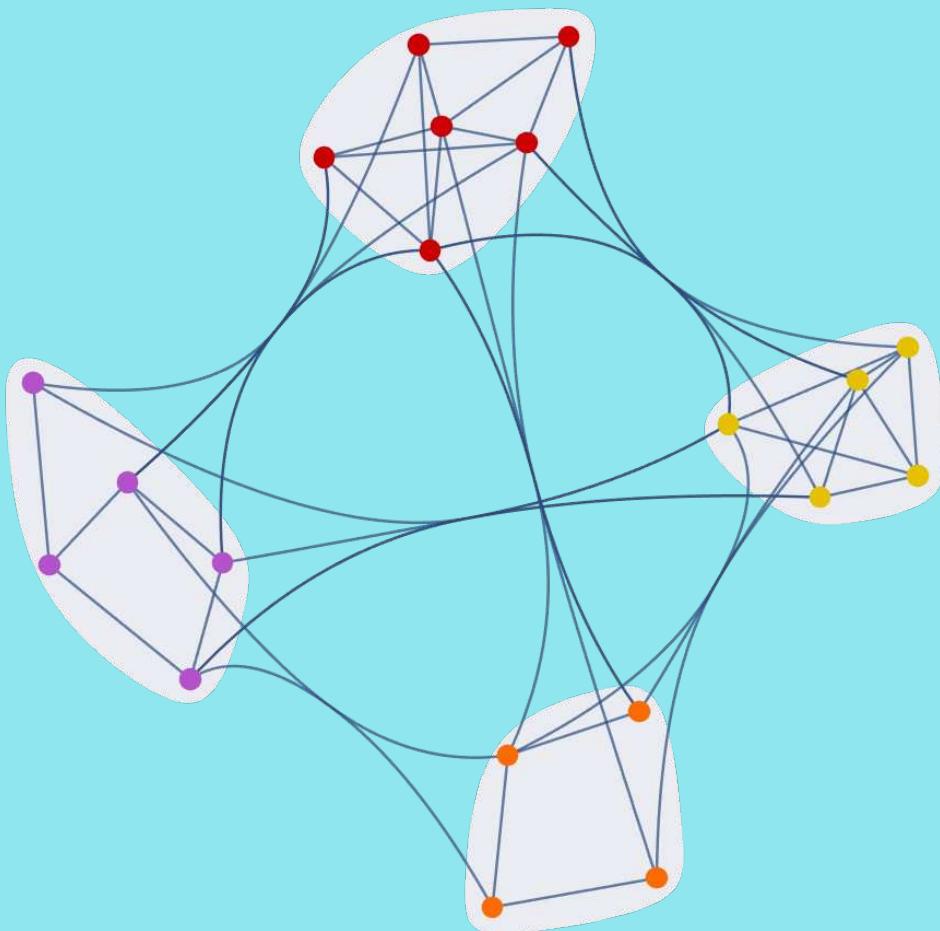


Widzimy, że drzewo Steinera S (czerwone krawędzie) zawiera wierzchołki terminalowe T (niebieskie) oraz dodatkowy wierzchołek 4. Jest to korzystne, bo droga z 1 do 2 jest krótsza prowadząc przez 4, gdzie $0.3+0.5=0.8$, niż idąc bezpośrednio, gdzie długość wynosi 1.

2.10 Zadania

1. Narysuj średniej wielkości (ok. 10 wierzchołków i 20-30 krawędzi) graf spójny G na kartce papieru. Przydziel etykiety wierzchołkom i „wprowadź graf do komputera”. Narysuj graf.
2. Znajdź minimalne drzewo spinające grafu G .
3. Przydziel wagi krawędziom grafu G i znajdź najkrótszą drogę między wybraną parą wierzchołków.
4. Potraktuj wagi krawędzi G jako przepustowości sieci i znajdź maksymalny przepływ między dwoma wierzchołkami.
5. Zbadaj własności grafu **nx.random_geometric_graph(200, d)** w zależności od $d \in (0, 1)$, tj. spójność, liczbę krawędzi, rozmiar największej kliki.
6. Narysuj na kartce paperu sieć (na grafie skierowanym) o kilkunastu wierzchołkach, zadając przepustowości krawędzi. Zastosuj algorytm **nx.maximum_flow** do znalezienia maksymalnego przepływu.

Podręcznik przedstawia w bardzo prosty sposób klasyczne zagadnienia matematyki dyskretnej. Wydanie drugie zawiera obszerny dodatek w formie Jupyter Book z przykładami i zadaniami dot. poruszanych problemów. Programy, napisane w języku Python jako Jupyter Notebook, mogą być bezpośrednio (bez instalacji czy konfiguracji) wykonywane przez czytelnika w chmurze obliczeniowej.



ISBN 978-83-962099-1-7

9 788396 209917