OPTYMALIZACJA KOMBINATORYCZNA - KOMPENDIUM

Kompendium dedykuję wszystkim forumowym hejterom. Pierdolta się!

NAJMNIEJSZE ODLEGŁOŚCI MIĘDZY PARAMI WIERZCHOŁKÓW

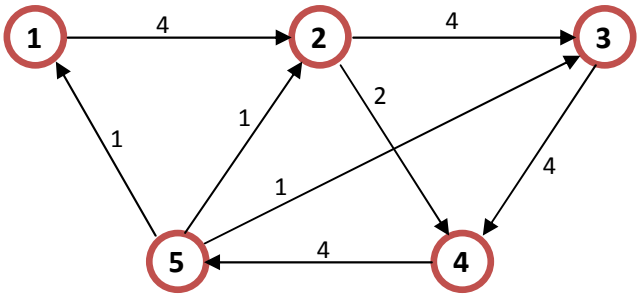
Do wyznaczania najmniejszej odległości pomiędzy parami wierzchołków (przy korzystaniu z wierzchołków pośrednich) służy algorytm Floyda. Graf musi być skierowany i nie posiadać cykli o ujemnej długości. Warunek nieujemności cyklu wynika z tego, że w grafie o ujemnych cyklach najmniejsza odległość między niektórymi wierzchołkami jest nieokreślona, ponieważ zależy od liczby przejść w cyklu. Mając dane odległości pomiędzy niektórymi wierzchołkami, możemy znaleźć odległości do reszty z nich, jeśli tylko są one z danego wierzchołka osiągalne.

//można by dodać trochę teorii + wzór

//Jeśli w danej iteracji, ułożenie pustych komórek w obszarze aktualnie zaznaczonym uniemożliwia nam jakiekolwiek próby poprawy wartości (np. cały zaznaczony wiersz jest pusty, oprócz oczywiście zera na głównej przekątnej), przerywamy algorytm, gdyż na skutek braku dalszych połączeń w grafie, nie poprawimy żadnych wyliczonych odległości. //do weryfikacji

PRZYKŁAD 1

Wyznacz najmniejsze odległości pomiędzy wszystkimi wierzchołkami w grafie.



Rysujemy tabelę z odległościami, jakie mamy dane w treści zadania.

	1	2	3	4	5
1	0	4			
2		0	4	2	
3			0	4	
4				0	4
5	1	1	1		0

Przyjmujemy zerową odległość wierzchołka do samego siebie. Jeśli nie ma bezpośredniego połączenia między dwoma wierzchołkami (w danym kierunku), ich odległość od siebie przyjmujemy jako nieskończoność i oznaczamy pustą komórką w tabeli.

Przykładowo, odległość od 1 do 2 wynosi 4, zatem w komórce [1, 2] umieszczamy wartość 4. (nie w [2, 1], bo tabelę czytamy w kolejności wiersz → kolumna, pamiętajmy, że jest to graf skierowany)

Przeprowadzamy pierwszą iterację algorytmu. Zaznaczamy pierwszy wiersz i pierwszą kolumnę. k = 1

	1	2	3	4	5
1	0	4			
2		0	4	2	
3			0	4	
4				0	4
5	1	1	1		0

Wartości w zaznaczonym wierszu i kolumnie nie zmieniają się w danej iteracji algorytmu. Zaznaczony obszar wypełniamy przepisując wartości z poprzedniego kroku algorytmu.

Nie rozpatrujemy w tym problemie komórek na głównej przekątnej tabeli – nie poprawimy w żaden sposób odległości wierzchołka do samego siebie, jeśli nie mamy cykli ujemnych w grafie. //?????

Patrzymy zawsze na wartości z poprzedniej iteracji, nowe wartości wpisujemy w aktualnej tabeli. Startujemy od pustej tabeli z wyzerowaną główną przekątną i w zależności od algorytmu, albo przepisujemy poprzednią wartość komórki, albo poprawiamy ją na lepszą.

Algorytm działa następująco: przechodzimy po wszystkich niezaznaczonych komórkach – są one usytuowane jak widać, na przecięciach komórek zaznaczonych. Przykładowo, komórka [5, 2] znajduje się na przecięciu komórek [5, 1] i [1, 2]. Jeśli wartości w obu zaznaczonych komórkach tworzących przecięcie jest różna od nieskończoności, sumujemy te dwie wartości i jeśli dają mniejszą liczbę niż ta, która jest w komórce na przecięciu, poprawiamy wartość tej komórki. Dla powyższych komórek, wartość [5, 1] + [1, 2] = 1 + 4 = 5. Natomiast istniejąca już wartość [5, 2] wynosi 1, zatem jest lepsza od wyliczonej – nic nie zmieniamy.

W tej iteracji była to jedyna komórka, którą ewentualnie mogliśmy poprawić, wszystkie inne zaznaczone komórki mają wartość nieskończoności, czyli na pewno nie poprawią nam żadnej odległości. Przepisujemy pozostałe wartości z poprzedniej iteracji (nie udało się nam ich poprawić).

Iteracja druga. Zaznaczamy drugi wiersz i drugą kolumnę. k = 2

	1	2	3	4	5
1	0	4	8	6	
2		0	4	2	
3			0	4	
4				0	4
5	1	1	1	3	0

W tej iteracji, nie zmieniają się wartości w drugim wierszu i drugiej kolumnie. Przepisujemy do nich wartości z poprzedniej iteracji.

Przechodźmy niezaznaczone komórki wierszami w kolejności od lewej do prawej, patrząc na tabelę z poprzedniej iteracji, wprowadzając zmiany w tabeli aktualnej.

Pierwszą komórką w której możemy poprawić wartość, jest [1, 3]. Jak widać, obydwie zaznaczone

komórki mają wartości różne od nieskończoności. Zatem [1, 2] + [2, 3] = 4 + 4 = 8, komórka [1, 3] jest pusta, ma wartość nieskończoności, czyli wartość 8 poprawia nam odległość, wpisujemy 8 do komórki [1, 3]. Idziemy dalej, [1, 2] + [2, 4] = 4 + 2 = 6, jest to wartość lepsza niż nieskończoność, wpisujemy 6 do komórki [1, 4]. Gdybyśmy chcieli poprawić również wartość komórki [1, 5], zauważamy, że [2, 5] ma wartość nieskończoności i nie ma w związku z tym na to szans.

Przechodzimy po kolejnych niezaznaczonych komórkach i okazuje się, że dopiero w [5, 4] możemy starać się poprawić wartość. Będzie to [5, 2] + [2, 4] = 1 + 2 = 3, w porównaniu do nieskończoności, wartość 3 jest oczywiście lepsza, wpisujemy zatem 3 do komórki [5, 4].

Iteracja trzecia. Zaznaczamy trzeci wiersz i trzecią kolumnę. k = 3

	1	2	3	4	5
1	0	4	8	6	
2		0	4	2	
3			0	4	
4				0	4
5	1	1	1	3	0

Jak widać po zawartości komórek [3, 1] i [3, 2], nie mamy czego szukać w całej lewej części tabeli, nieważne co stanowi wartość drugiej „współrzędnej”, wyniku nie poprawimy. Pamiętajmy, że uzupełniając (poprawiając) aktualną tabelę, patrzymy na tabelę z poprzedniej iteracji. Jeśli poprawiliśmy wartość, wpisujemy ją do tabeli, jeśli nie poprawiliśmy, przepisujemy wartość poprzednią.

[1, 3] + [3, 4] = 8 + 4 = 12 jest większe niż 6 = [1, 4], zatem wyniku nie poprawia, przepisujemy.
[2, 3] + [3, 4] = 4 + 4 = 8 jest większe niż 2 = [2, 4], przepisujemy starą wartość jak wyżej.
[5, 3] + [3, 4] = 1 + 4 = 5 jest większe niż 3 = [5, 4], również nic nie zmieniamy, jak poprzednio.

Iteracja czwarta. Zaznaczamy czwarty wiersz i czwartą kolumnę. k = 4

	1	2	3	4	5
1	0	4	8	6	10
2		0	4	2	6
3			0	4	8
4				0	4
5	1	1	1	3	0

Znów cała lewa część tabeli odpada, za to w prawej pojawią się na pewno nowe wartości.

[1, 4] + [4, 5] = 6 + 4 = 10 jest mniejsze od nieskończoności, poprawiamy.
[2, 4] + [4, 5] = 2 + 4 = 6 jak wyżej.
[3, 4] + [4, 5] = 4 + 4 = 8 tak samo.

Iteracja piąta i ostatnia. Zaznaczamy piąty wiersz i piątą kolumnę. k = 5

	1	2	3	4	5
1	0	4	8	6	10
2	7	0	4	2	6
3	9	9	0	4	8
4	5	5	5	0	4
5	1	1	1	3	0

Tym razem mamy szansę na poprawienie wszystkich możliwych komórek (oprócz zer na głównej przekątnej), z uwagi na różne od nieskończoności wartości we wszystkich „współrzędnych” na zaznaczonym obszarze.

[5, 2] + [1, 5] = 1 + 10 = 11 jest większe niż 4 = [1, 2], przepisujemy starą wartość.
[5, 3] + [1, 5] = 1 + 10 = 11 jest większe niż 8 = [1, 3], przepisujemy.
[5, 4] + [1, 5] = 3 + 10 = 13 jest większe niż 6 = [1, 4], przepisujemy.

[5, 1] + [2, 5] = 1 + 6 = 7 jest mniejsze od nieskończoności, poprawiamy wartość w [2, 1] na 7.
[5, 3] + [2, 5] = 1 + 6 = 7 jest większe niż 4 = [2, 3], przepisujemy starą wartość.
[5, 4] + [2, 5] = 3 + 6 = 9 jest większe niż 2, przepisujemy.

[5, 1] + [3, 5] = 1 + 8 = 9 jest mniejsze od nieskończoności, poprawiamy wartość w [3, 1] na 9.
[5, 2] + [3, 5] = 1 + 8 = 9 jest mniejsze od nieskończoności, poprawiamy wartość w [3, 2] na 9.
[5, 4] + [3, 5] = 3 + 8 = 11 jest większe niż [3, 4], przepisujemy starą wartość.

[5, 1] + [4, 5] = 1 + 4 = 5 jest mniejsze od nieskończoności, poprawiamy wartość w [4, 1] na 5.
[5, 2] + [4, 5] = 1 + 4 = 5 jest mniejsze od nieskończoności, poprawiamy wartość w [4, 2] na 5.
[5, 3] + [4, 5] = 1 + 4 = 5 jest mniejsze od nieskończoności, poprawiamy wartość w [4, 1] na 5.

Wyznaczyliśmy w ten sposób najkrótsze odległości między wszystkimi parami wierzchołków w grafie.

W praktyce, licząc zadanie na kartce, dla kolejnych iteracji rysujemy pustą tabelę z wyzerowaną główną przekątną a do zaznaczonego aktualnie obszaru przepisujemy wartości z poprzedniej iteracji. Dalej sprawdzamy poprzednie wartości w niezaznaczonych komórkach, przepisując je do aktualnej tabeli lub poprawiając na lepsze. Innym sposobem jest przepisanie całej poprzedniej tabeli do aktualnej iteracji i poprawianie wartości tylko w obrębie aktualnej tabeli – wiąże się to z ewentualnym skreślaniem wartości na kartce, ale trudniej o pomyłkę. Wybór sposobu zależy od osobistych preferencji.

PRZYKŁAD 2

Dla zadanej macierzy grafu, wyznacz najkrótsze ścieżki między wszystkimi parami wierzchołków.

	1	2	3	4	5	6
1	0			1		
2		0	1			
3	2		0	3		
4			4	0		2
5		2	5		0	
6					1	0

Mamy gotową tabelę z odległościami, przystępujemy zatem do kolejnych iteracji algorytmu. Pamiętajmy, że wartości w zaznaczonym aktualnie wierszu i kolumnie nie zmieniają się, według nich staramy się poprawić wartości niezaznaczonych komórek na mniejsze. Patrzymy na tabelę z poprzedniej iteracji i w aktualnej tabeli poprawiamy wartości lub przepisujemy stare. W poniższych obliczeniach, pierwsza liczba pochodzi ze „współrzędnej" wiersza, druga kolumny. Porównujemy oczywiście ich sumę z wartością na przecięciu „współrzędnych".

	1	2	3	4	5	6
1	0			1		
2		0	1			
3	2		0	3		
4			4	0		2
5		2	5		0	
6					1	0

k = 1

2 + 1 ≥ 3 (przepisujemy 3)
(przepisujemy resztę)

	1	2	3	4	5	6
1	0			1		
2		0	1			
3	2		0	3		
4			4	0		2
5		2	3		0	
6					1	0

k = 2

2 + 1 < 5 (poprawiamy na 3)

	1	2	3	4	5	6
1	0			1		
2		0	1			
3	2		0	3		
4	6		4	0		2
5	5	2	3	6	0	
6					1	0

k = 3

1 + 2 < ∞
1 + 3 < ∞
4 + 2 < ∞
3 + 2 < ∞
3 + 3 < ∞

	1	2	3	4	5	6
1	0		5	1		3
2	3	0	1	4		6
3	2		0	3		5
4	6		4	0		2
5	5	2	3	6	0	8
6					1	0

k = 4

1 + 4 < ∞
1 + 2 < ∞
4 + 6 ≥ 3
4 + 4 ≥ 1
4 + 2 < ∞
3 + 6 ≥ 2
3 + 2 < ∞
6 + 6 ≥ 5
6 + 4 ≥ 3
6 + 2 < ∞

	1	2	3	4	5	6
1	0		5	1		3
2	3	0	1	4		6
3	2		0	3		5
4	6		4	0		2
5	5	2	3	6	0	8
6	6	3	4	7	1	0

k = 5

1 + 5 < ∞
1 + 2 < ∞
1 + 3 < ∞
1 + 6 < ∞

	1	2	3	4	5	6
1	0	6	5	1	4	3
2	3	0	1	4	7	6
3	2	8	0	3	6	5
4	6	5	4	0	3	2
5	5	2	3	6	0	8
6	6	3	4	7	1	0

k = 6

3 + 3 < ∞
3 + 4 ≥ 5
3 + 7 ≥ 1
3 + 1 < ∞
6 + 6 ≥ 3
6 + 4 ≥ 1
6 + 7 ≥ 4
6 + 1 < ∞
5 + 6 ≥ 2
5 + 3 < ∞
5 + 7 ≥ 3
5 + 1 < ∞
2 + 6 ≥ 6
2 + 3 < ∞
2 + 4 ≥ 4
2 + 1 < ∞
8 + 6 ≥ 5
8 + 3 ≥ 2
8 + 4 ≥ 3
8 + 7 ≥ 6

DOMKNIĘCIE PRZECHODNIE RELACJI BINARNEJ

Mając dany graf skierowany, możemy sprawdzić, czy wierzchołki są ze sobą połączone na zasadzie relacji przechodniości. Aby to ustalić, szukamy przechodnich ścieżek za pomocą wyznaczania najkrótszych dróg w grafie. Nadajemy wagi „1" każdej krawędzi i jeśli otrzymamy ∞ to dane węzły są niezależne.

//dodać teorię i wzory, przykład niepotrzebny

NAJBARDZIEJ NIEZAWODNE POŁĄCZENIA

Problem wyznaczania prawdopodobieństwa najbardziej niezawodnych połączeń można rozwiązać również algorytmem Floyda, odpowiednio modyfikując odległości między wierzchołkami w grafie. Wypełniając tabelę początkową dla wyznaczania odległości między wierzchołkami, stosujemy wzór:

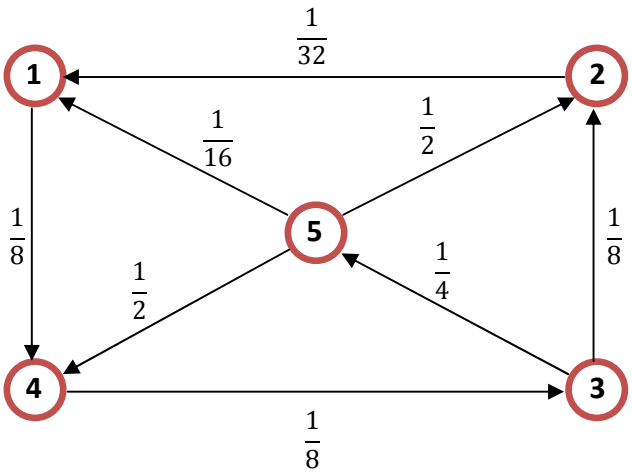
$$d(u, v) = -\log_{\beta} p(u, v)$$

Podstawa logarytmu β zależy od wartości zadanych prawdopodobieństw, należy ją dobrać tak, aby otrzymać sensowne wartości w tabeli odległości dla algorytmu Floyda.

//jak to do końca jest? Dobieramy podstawę według danych, żeby sensownie się liczyło tak?

PRZYKŁAD

Wyznacz prawdopodobieństwo najbardziej niezawodnego połączenia między wszystkimi parami wierzchołków w sieci, przy zadanych prawdopodobieństwach niezawodnej pracy poszczególnych par łącz.



	1	2	3	4	5
1	1			$\frac{1}{8}$	
2	$\frac{1}{32}$	1			
3		$\frac{1}{8}$	1		$\frac{1}{4}$
4			$\frac{1}{8}$	1	
5	$\frac{1}{16}$	$\frac{1}{2}$		$\frac{1}{2}$	1

Daną macierz prawdopodobieństw musimy zmodyfikować, abyśmy mogli jej użyć w algorytmie Floyda, zgodnie ze wzorem:

$$d(u,v) = -\log_\beta p(u,v)$$

Odległość początkową w tabeli Floyda obliczamy z logarytmu z danego prawdopodobieństwa niezawodnej pracy między łączami i biorąc wartość przeciwną. Mianowniki ułamków jak widać stanowią potęgi dwójki, stosujemy zatem logarytm o podstawie 2. Przeliczamy wszystkie wartości z macierzy wejściowej i umieszczamy je w nowej tabeli.

$d(1,4) = -\log_\beta p(1,4) = -\log_2 \frac{1}{8} = -(-3) = 3$, w miejsce wszystkich prawdopodobieństw $\frac{1}{8}$ wpisujemy odległość 3.
$d(2,1) = -\log_\beta p(2,1) = -\log_2 \frac{1}{32} = -(-5) = 5$, w miejsce wszystkich prawdopodobieństw $\frac{1}{32}$ wpisujemy odległość 5.
$d(3,5) = -\log_\beta p(3,5) = -\log_2 \frac{1}{4} = -(-2) = 2$, w miejsce wszystkich prawdopodobieństw $\frac{1}{4}$ wpisujemy odległość 2.
$d(5,1) = -\log_\beta p(5,1) = -\log_2 \frac{1}{16} = -(-4) = 4$, w miejsce wszystkich prawdopodobieństw $\frac{1}{16}$ wpisujemy odległość 4.
$d(5,2) = -\log_\beta p(5,2) = -\log_2 \frac{1}{2} = -(-1) = 1$, w miejsce wszystkich prawdopodobieństw $\frac{1}{2}$ wpisujemy odległość 1.

Wnikliwy czytelnik zauważy, że nie możemy przepisać jedynek z głównej przekątnej do tabeli algorytmu Floyda, oznaczałoby to, że odległość węzłów od samych siebie jest różna od zera. Zauważmy, że prawdopodobieństwo niezawodności łącza między samym sobą wynosi 1, czyli łącze jest w 100% niezawodne. Dla głównej przekątnej mamy wartość $-\log_2 1 = 0$, przenosząc to na odległości między węzłami, 100% niezawodności łącza tłumaczymy jako zerową odległość węzłów. Czym to prawdopodobieństwo jest mniejsze, tym odległości się bardziej zwiększają. Szukamy największego prawdopodobieństwa, czyli najmniejszej odległości, a do tego bardzo dobrze nadaje się algorytm Floyda. Zamieniamy jeszcze jedynki z wejściowej macierzy na zera w tabeli odległości i pierwszy krok za nami.

	1	2	3	4	5
1	0			3	
2	5	0			
3		3	0		2
4			3	0	
5	4	1		1	0

Otrzymaliśmy tabelę wejściową dla algorytmu Floyda, wyznaczania najkrótszych dróg w grafie.

Jak pamiętamy, większa odległość oznacza mniejsze prawdopodobieństwo niezawodności łącza. Wyliczmy odległości między wszystkimi parami wierzchołków, tym razem już bez szczegółowego objaśniania kolejnych kroków.

k = 1

	1	2	3	4	5
1	0			3	
2	5	0		8	
3		3	0		2
4			3	0	
5	4	1		1	0

k = 2

	1	2	3	4	5
1	0			3	
2	5	0		8	
3	8	3	0	11	2
4			3	0	
5	4	1		1	0

k = 3

	1	2	3	4	5
1	0			3	
2	5	0		8	
3	8	3	0	11	2
4	11	6	3	0	5
5	4	1		1	0

k = 4

	1	2	3	4	5
1	0	9	6	3	8
2	5	0	11	8	13
3	8	3	0	11	2
4	11	6	3	0	5
5	4	1	4	1	0

k = 5

	1	2	3	4	5
1	0	9	6	3	8
2	5	0	11	8	13
3	6	3	0	3	2
4	9	6	3	0	5
5	4	1	4	1	0

Otrzymaliśmy zatem najmniejsze odległości między parami wszystkich wierzchołków.

Mając wyliczone odległości, możemy przekształcić je na prawdopodobieństwa niezawodności połączeń, postępując odwrotnie niż poprzednio. Stosujemy wzór:

$$p(u,v) = \beta^{-d(u,v)}$$

Gdzie β to wybrana przez nas podstawa logarytmowania. Mamy zatem:

$$p(1,1) = 2^{-d(1,1)} = 2^{-0} = 1$$
$$p(1,2) = 2^{-d(1,2)} = 2^{-9} = \frac{1}{512}$$
$$p(1,3) = 2^{-d(1,3)} = 2^{-6} = \frac{1}{64}$$
$$p(1,4) = 2^{-d(1,4)} = 2^{-3} = \frac{1}{8}$$
$$p(1,5) = 2^{-d(1,5)} = 2^{-8} = \frac{1}{256}$$
$$\vdots$$
$$p(5,4) = 2^{-d(5,4)} = 2^{-1} = \frac{1}{2}$$
$$p(5,5) = 2^{-d(5,5)} = 2^{-0} = 1$$

	1	2	3	4	5
1	1	$\frac{1}{512}$	$\frac{1}{64}$	$\frac{1}{8}$	$\frac{1}{256}$
2	$\frac{1}{32}$	1	$\frac{1}{2048}$	$\frac{1}{256}$	$\frac{1}{8192}$
3	$\frac{1}{64}$	$\frac{1}{8}$	1	$\frac{1}{8}$	$\frac{1}{4}$
4	$\frac{1}{512}$	$\frac{1}{64}$	$\frac{1}{8}$	1	$\frac{1}{32}$
5	$\frac{1}{16}$	$\frac{1}{2}$	$\frac{1}{16}$	$\frac{1}{2}$	1

Ostatecznie otrzymujemy macierz prawdopodobieństw niezawodnego połączenia między wszystkimi parami węzłów (o ile między wszystkimi istnieją połączenia, macierz jest w całości wypełniona).

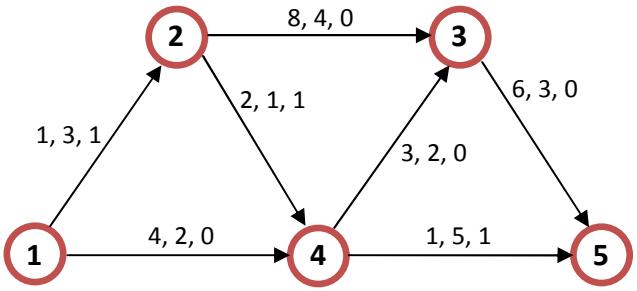
Przechodząc po krawędziach grafu między kolejnymi wierzchołkami, prawdopodobieństwa niezawodności łącz oczywiście mnożymy przez siebie. Początkowe przekształcenie prawdopodobieństw na odległości pozwoliło nam wykorzystać algorytm Floyda do rozwiązania problemu. Jak widać, uzyskane wyniki pokrywają się z intuicyjnym rozwiązaniem zadania poprzez przechodzenie grafu.

PRZEPŁYW O MINIMALNYM KOSZCIE

Teoria

PRZYKŁAD 1

Wyznacz koszty najtańszych ścieżek między wszystkimi parami wierzchołków w grafie dla problemu wyznaczania przepływu o minimalnym koszcie.



koszt, pojemność, przepływ

Dla danego grafu, kosztów przepływu jednej jednostki medium, pojemności łącz i ich aktualnego przepływu, mamy znaleźć minimalny koszt przepływu jednej jednostki pomiędzy wszystkimi parami węzłów. Ponownie problem możemy sprowadzić do poszukiwania najkrótszych ścieżek w grafie, najpierw należy jednak odpowiednio zmodyfikować wagi krawędzi grafu.

Na wagi łuków składają się trzy liczby. Pierwsza oznacza koszt przesłania jednej jednostki medium danym łączem, w kierunku wskazywanym przez zwrot łuków. Druga liczba to pojemność łącza, czyli maksymalna liczba jednostek nim płynąca. Ostatnia liczba wskazuje na przepływ, czyli liczbę jednostek aktualnie w nim płynącą. Przykładowo, wagi łuku 1 → 4 mówią nam, że koszt przesłania jednej jednostki tym łączem wynosi 4, mogą nim płynąć maksymalnie 2 jednostki, aktualnie nie płynie nic.

Przepływ jest nasycony, jeśli danym łączem przepływa maksymalna liczba jednostek. Przepływ jest zerowy, jeśli nic nie płynie.

Posiadane informacje musimy zamienić na odległości między parami wierzchołków w grafie, aby skorzystać z algorytmu Floyda.

Rozpatrujemy wszystkie bezpośrednie połączenia między węzłami. Standardowo wypełniając tabelę dla Floyda, wpisujemy odległość z wierzchołka **i** do **j** w komórce [i, j], pod warunkiem, że istnieje krawędź i → j. Tym razem wypełniamy obydwie komórki, [i, j] oraz [j, i], ponieważ istnieje bezpośrednie połączenie między nimi. Jaką wartością? Według poniższych zasad:

Rozpatrując komórkę [i, j] w tablicy odległości wierzchołków, wpisujemy do niej:
• koszt przepływu jednostki, jeśli istnieje nienasycony (może być zerowy) przepływ i → j
• minus koszt przepływu jednostki, jeśli istnieje niezerowy (może być nasycony) przepływ j → i
• ∞ w pozostałych przypadkach
Za każdym razem, w miarę możliwości, poprawiając wartości na mniejsze.

Uzupełnijmy tabeli dla algorytmu Floyda według powyższych reguł. Patrzymy na oryginalny graf, przechodzimy po tabeli wierszami od lewej do prawej. Na początku zerujemy główną przekątną w naszej tabeli, nie sprawdzamy wartości na głównej przekątnej, bo nie poprawimy odległości wierzchołków od samych siebie, jeśli nie mamy cykli ujemnych. //??????????

	1	2	3	4	5
1	0	1			
2		0			
3			0		
4				0	
5					0

Szukamy połączenia 1 – 2. Istnieje między nimi łuk (1 → 2), zatem możemy je dalej rozpatrywać. Łuk jest skierowany od wierzchołka 1 do wierzchołka 2, ma koszt równy 1, pojemność 3 i przepływ 1. Sprawdzamy zatem powyższe warunki:

pojemność(1, 2) – przepływ(1, 2) = 3 – 1 = 2 ≠ 0, przepływ jest nienasycony, spełniony został pierwszy warunek, w komórce [1, 2] wpisujemy wartość koszt(1, 2) = 1.

Drugiego warunku nie mamy możliwości sprawdzić, bo nie ma łuku prowadzącego w przeciwnym kierunku (2 → 1), uzyskana wartość jest zatem najmniejszą możliwą i taką zostawiamy. W tym przykładzie nie zajdzie sytuacja poprawiania kosztu na mniejszy, z uwagi na brak komplementarnych łuków między wierzchołkami. Przechodzimy zatem do następnej komórki.

Szukamy połączenia 1 – 3. Nie istnieje między nimi łuk. Przechodzimy dalej.

	1	2	3	4	5
1	0	1		4	
2		0			
3			0		
4				0	
5					0

Połączenie 1 – 4 istnieje. Stanowi ono łuk 1 → 4 o wagach 4, 2, 0. Sprawdźmy warunki:

pojemność(1, 4) – przepływ(1, 4) = 2 – 0 = 2 ≠ 0, przepływ jest nienasycony, spełniony został pierwszy warunek, w komórce [1, 4] wpisujemy wartość koszt(1, 4) = 4.

Nie ma łuku przeciwnego 4 → 1, pomijamy sprawdzanie drugiego warunku. Przechodzimy dalej.

Połączenie 1 – 5 nie istnieje. Idziemy do drugiego wiersza.

	1	2	3	4	5
1	0	1		4	
2	-1	0			
3			0		
4				0	
5					0

Połączenie 2 – 1 istnieje mimo, że nie ma łuku 2 → 1. Istnieje natomiast łuk 1 → 2, sprawdzamy zatem drugi warunek, dla przepływu płynącego w przeciwnym kierunku niż łuk powstały z aktualnie rozpatrywanych indeksów tabeli (rozpatrujemy, czy dodać do tabeli łuk 2 → 1, mimo, że taki w grafie nie istnieje – wspomnieliśmy już wcześniej, że tym razem liczy się sam fakt połączenia, nie jego kierunek, aby dodać łuk do tablicy wzajemnej odległości wierzchołków //???????????):

przepływ(1, 2) ≠ 0, zatem wpisujemy do komórki [2, 1] wartość –koszt(1, 2) = -1.

	1	2	3	4	5
1	0	1		4	
2	-1	0	8		
3			0		
4				0	
5					0

Połączenie 2 – 3 istnieje dzięki łukowi 2 → 3, jego zwrot jest zgodny ze sprawdzanymi współrzędnymi w tabeli, więc sprawdzamy warunek pierwszej reguły:

pojemność(2, 3) – przepływ(2, 3) = 4 – 0 = 4 ≠ 0, mamy nienasycony przepływ, wpisujemy zatem do komórki [2, 3] wartość koszt(2, 3) = 8.

Połączenie 2 – 4 istnieje, jest to łuk 2 → 4, sprawdzamy pierwszy warunek:

pojemność(2, 4) – przepływ(2, 4) = 1 – 1 = 0, przepływ jest nasycony, nie możemy na tej podstawie poprawić wartości w komórce [2, 4] (nie puścimy tym łukiem nic więcej w jego kierunku, bo łącze jest już pełne, dla algorytmu Floyda oznacza to, że tej krawędzi w ogóle w grafie nie ma).

Przechodzimy dalej. Połączenie 2 – 5 nie istnieje, przechodzimy do trzeciego wiersza tabeli.

Połączenie 3 – 1 nie istnieje.

Połączenie 3 – 2 istnieje, jednak za sprawą łuku 2 → 3 czyli w przeciwnym kierunku niż rozpatrywany. Jak pamiętamy, w związku z tym sprawdzamy drugą regułę:

przepływ(2, 3) = 0, nie możemy na tej podstawie poprawić wartości w komórce [3, 2] (gdyby przepływ 2 → 3 był niezerowy, moglibyśmy puścić coś w kierunku 3 → 2 poprzez zmniejszenie przepływu 2 → 3, czyli zamiast puszczać coś od 3 do 2, zmniejszamy dostawy z 2 do 3 – efekt ten sam). Zostawiamy puste pole i idziemy dalej.

Połączenie 3 – 4 istnieje, również w kierunku przeciwnym 4 → 3, zatem druga reguła:

przepływ(4, 3) = 0, nie możemy poprawić wartości w komórce [3, 4].

	1	2	3	4	5
1	0	1		4	
2	-1	0	8		
3			0		6
4				0	
5					0

Połączenie 3 – 5 istnieje, mamy łuk 3 → 5, sprawdzamy pierwszy warunek:

pojemność(3, 5) – przepływ(3, 5) = 3 – 0 = 3 ≠ 0, przepływ nienasycony, wpisujemy do komórki [3, 5] wartość koszt(3, 5) = 6.

Przechodzimy do czwartego wiersza w tabeli.

Połączenie 4 – 1 istnieje, mamy jednak łuk odwrotny 1 → 4, sprawdzamy drugi warunek:

przepływ(1, 4) = 0, nie możemy poprawić wartości w komórce [4, 1].

	1	2	3	4	5
1	0	1		4	
2	-1	0	8		
3			0		6
4		-2		0	
5					0

Połączenie 4 – 2 również w kierunku przeciwnym, sprawdzamy:

przepływ(2, 4) = 1 ≠ 0, przepływ niezerowy, możemy poprawić wartość [4, 2] na –koszt(2, 4) = -2.

	1	2	3	4	5
1	0	1		4	
2	-1	0	8		
3			0		6
4		-2	3	0	
5					0

Połączenie 4 – 3 idzie w kierunku zgodnym ze sprawdzanym aktualnie łukiem w tablicy odległości.

pojemność(4, 3) – przepływ(4, 3) = 2 – 0 = 2 ≠ 0, do komórki [4, 3] trafia koszt(4, 3) = 3.

	1	2	3	4	5
1	0	1		4	
2	-1	0	8		
3			0		6
4		-2	3	0	1
5					0

Połączenie 4 – 5 również w zgodnym kierunku, sprawdzamy:

pojemność(4, 5) – przepływ(4, 5) = 5 – 1 = 4 ≠ 0, do komórki [4, 5] trafia koszt(4, 5) = 1.

Połączenia 5 – 1 i 5 – 2 nie istnieją.

Połączenie 5 – 3 jest postaci 3 → 5, sprawdzamy drugi warunek:

przepływ(3, 5) = 0, nie poprawimy dotychczasowej wartości w [5, 3].

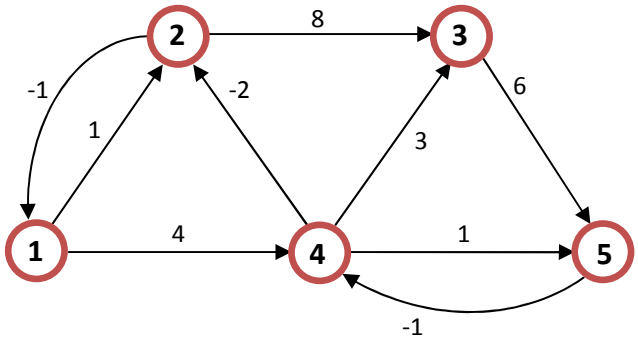
	1	2	3	4	5
1	0	1		4	
2	-1	0	8		
3			0		6
4		-2	3	0	1
5				-1	0

Połączenie 5 – 4 jest postaci 4 → 5, sprawdzamy drugi warunek:

przepływ(4, 5) = 1 ≠ 0, poprawiamy dotychczasową wartość [5, 4] jeśli wartość –koszt(4, 5) jest lepsza.

Uzyskaliśmy w powyższy sposób macierz odległości pomiędzy wierzchołkami w grafie, reprezentuje ona tak naprawdę koszty przesłania jednej jednostki medium między tymi wierzchołkami. Oczywiście im większa odległość tym przesłanie medium ma większy koszt.

Zauważmy, że w powyższej tabeli występują ujemne odległości – może się to wydawać pozbawione sensu, ale przekładając to na ujemne koszty, uzyskujemy sensowny problem. Można wyobrazić sobie sytuację, że przesyłając medium danym kanałem w określonym kierunku, zarabiamy zamiast za przesył płacić. Graf powstały z uzyskanych odległości nie posiada cykli o ujemnej długości, możemy więc wyliczyć najkrótsze ścieżki w grafie za pomocą algorytmu Floyda.



Dla uzyskanej tabeli, wyliczamy zatem odległości pomiędzy wszystkimi parami wierzchołków w grafie. Wynik obliczeń (bez szczegółowych wyjaśnień) przedstawia poniższa tabela.

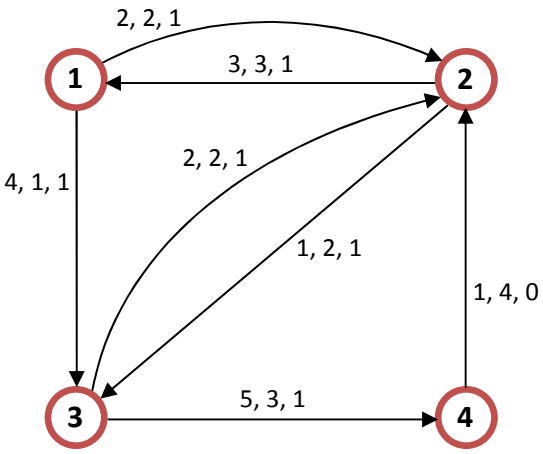
	1	2	3	4	5
1	0	1	7	4	5
2	-1	0	6	3	4
3	2	3	0	5	6
4	-3	-2	3	0	1
5	-4	-3	2	-1	0

Metoda pozostaje ta sama, należy tylko pamiętać o możliwości powstawania ujemnych liczb podczas sumowania wartości ze „współrzędnych” zaznaczonych obszarów w kolejnych iteracjach – jak zwykle, poprawiamy istniejące wartości na jak najmniejsze.

Otrzymaliśmy niniejszym minimalne koszty przepływu jednej jednostki medium pomiędzy wszystkimi parami wierzchołków.

PRZYKŁAD 2

Wyznacz koszty najtańszych ścieżek między wszystkimi parami wierzchołków w grafie dla problemu wyznaczania przepływu o minimalnym koszcie.



koszt, pojemność, przepływ

W tym przykładzie mamy cykle pomiędzy wierzchołkami, zobaczmy jak wpływa to na przełożenie przepływów na odległości między węzłami w celu wyznaczenia najkrótszych ścieżek między nimi (de facto kosztów przepływów).

Przygotujmy tabelę z odległościami na podstawie danego grafu. Tym razem bez szczegółowego objaśniania wszystkich kroków oraz przechodzenia po całej tabeli – pomijamy w naszych rozważaniach nieistniejące połączenia. Zerujemy główną przekątną macierzy odległości i przechodzimy ją wierszami od lewej do prawej.

	1	2	3	4
1	0	-3		
2		0		
3			0	
4				0

Połączenie 1 – 2. Mamy łuk wychodzący i wchodzący do wierzchołka nr 1. Wiemy zatem, że musimy sprawdzić obydwa warunki reguły, według której ustalamy wartości do wpisania w tabeli obok.

Mamy łuk 1 → 2 (zgodny z kierunkiem sprawdzanego połączenia), sprawdzamy warunek pierwszy. Przepływ jest nienasycony, możemy więc wpisać w komórce [1, 2] wartość 2 (czyli koszt(1, 2)).

Nie zapominajmy jednak o tym, że rozpatrywane połączenie składa się z jeszcze jednego łuku. Jest to oczywiście 2 → 1, biegnie w kierunku odwrotnym, sprawdzamy więc drugą regułę. Przepływ jest niezerowy, możemy zatem wartością -3 (–koszt(2, 1)) spróbować poprawić dotychczasową zawartość komórki [1, 2]. Oczywiście -3 < 2, zatem ostatecznie, na skutek rozpatrzenia wszystkich łuków stanowiących połączenie 1 – 2, do komórki [1, 2] wpisujemy wartość -3.

Połączenie 1 – 3. Przepływ 1 → 3 jest nasycony, warunek pierwszy niespełniony, nic nie zmieniamy.

	1	2	3	4
1	0	-3		
2	-2	0		
3			0	
4				0

Połączenie 2 – 1. Przepływ 2 → 1 jest nienasycony, do komórki [2, 1] trafia wartość 3 (koszt(2, 1)). Przepływ 1 → 2 jest niezerowy, jego wartość (–koszt(1, 2)) poprawia nam wartość 3 na -2.

	1	2	3	4
1	0	-3		
2	-2	0	-2	
3			0	
4				0

Połączenie 2 – 3. Przepływ 2 → 3 jest nienasycony, zatem do komórki [1, 2] wpisujemy 1. Przepływ 3 → 2 jest niezerowy, wartość wnoszona przez niego, poprawia nam wynik w komórce [1, 2] na -2.

Połączenie 2 – 4. Nie ma łuku 2 → 4, ale w niczym nam to nie przeszkadza, mamy za to łuk 4 → 2, który należy sprawdzić pod kątem spełniania znanych nam reguł. Jego kierunek jest odwrotny do połączenia, patrzymy więc na przepływ – zerowy przepływ oznacza, że łuk nie wnosi nic do rozwiązania.

	1	2	3	4
1	0	-3		
2	-2	0	-2	
3	-4		0	
4				0

Połączenie 3 – 1. Sytuacja podobna, sprawdzamy wartość przepływu 1 → 3. Jest on niezerowy, możemy zatem wpisać do komórki [3, 1] wartość -4.

	1	2	3	4
1	0	-3		
2	-2	0	-2	
3	-4	-1	0	
4				0

Połączenie 3 – 2. Łuki prowadzą w obu kierunkach, jak zwykle najpierw rozpatrujemy ten zgodny ze sprawdzanym połączeniem (zakładając, że pierwszy wierzchołek w połączeniu to początek łuku a drugi stanowi koniec – jest to oczywiście kwestia umowna i wprowadzona w celu dydaktycznym). Łuk 3 → 2 jest nienasycony, wpisujemy do [3, 2] wartość 2. Być może mamy jeszcze szansę na poprawienie tej wartości za sprawą drugiego istniejącego łuku. Sprawdźmy 2 → 3, przepływ w odwrotnym kierunku jest niezerowy, możemy poprawić poprzednią wartość 2 na -1 (oczywiście, gdyby ten przepływ był zerowy, poprzednia wartość 2 zostaje na swoim miejscu).

	1	2	3	4
1	0	-3		
2	-2	0	-2	
3	-4	-1	0	5
4				0

Połączenie 3 – 5 za sprawą nienasyconego łuku 3 → 5, pozwala nam bez wahania wpisać w komórce [3, 5] naszej tabeli, wpisać wartość 5.

	1	2	3	4
1	0	-3		
2	-2	0	-2	
3	-4	-1	0	5
4		1		0

Połączenie 4 – 2. Łuk 4 → 2 z kierunkiem zgodnym do rozpatrywanego, przepływ nienasycony, w komórce [4, 2] wpisujemy wartość 1.

	1	2	3	4
1	0	-3		
2	-2	0	-2	
3	-4	-1	0	5
4		1	-5	0

Połączenie 4 – 3 za sprawą łuku przeciwnego 3 → 4, którego przepływ jest niezerowy, pozwala nam na wpisanie do komórki [4, 3] wartości -5.

Pierwszy etap rozwiązania zadania mamy za sobą, powyższa tabela musi zostać jeszcze poddana działaniu algorytmu Floyda.

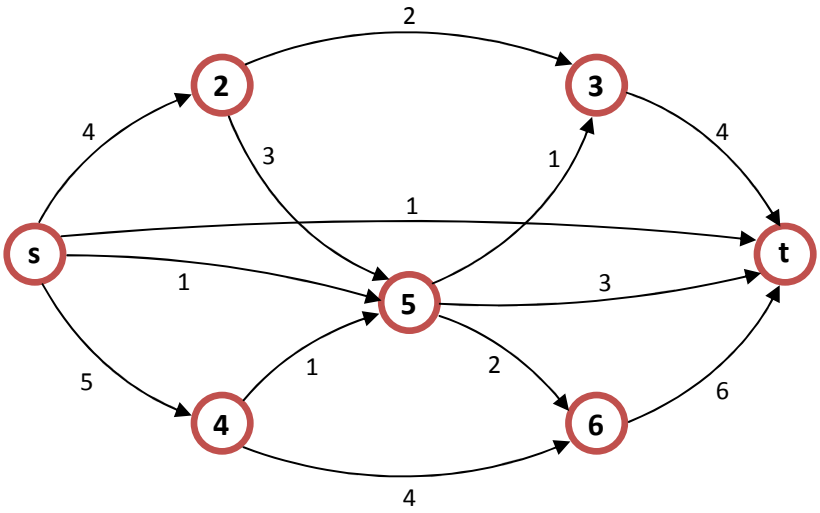
Rozrysowując graf odległości między węzłami w grafie, podobnie jak w pierwszym przykładzie, zauważymy istnienie cykli o ujemnej długości. //???????????? W przypadku zagadnienia odległości między wierzchołkami to nie ma sensu, ale czy tutaj też? Wydaje mi się, że chyba można to uznać za poprawne z akademickiego punktu widzenia, w praktyce pewnie by to nie wystąpiło. Nie mam innego przykładu niż ten, o którym wiem, że tabelka jest wypełniona dobrze a to chodzi, żeby to dobrze wytłumaczyć, nie o końcowy wynik

MAKSYMALNY PRZEPŁYW W SIECI

Teoria

PRZYKŁAD 1

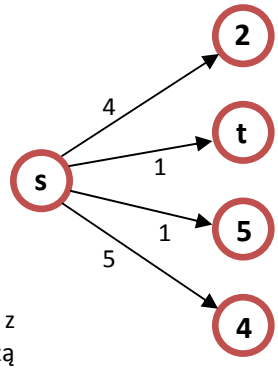
Znajdź maksymalny przepływ w poniższej sieci.



Etykiety reprezentują ograniczenia przepływu danymi łukami. Wierzchołek **s** oznacza źródło sieci, wierzchołek **t** – ujście. Zadanie polega na znalezieniu maksymalnego przepływu od źródła do ujścia jaki możliwy jest w tej sieci.

Do jego rozwiązania wykorzystujemy metodę Dinica, opierającą się na idei konstrukcji sieci warstwowych zawierających wszystkie najkrótsze ścieżki powiększające dany przepływ i nasycaniu tych ścieżek.

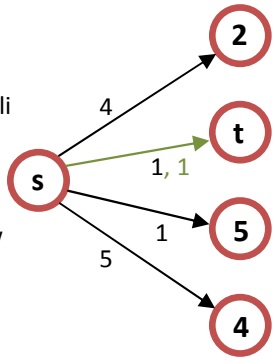
Sieci warstwowe to takie sieci, w których na kolejnych warstwach znajdują się wierzchołki osiągane bezpośrednio z warstwy poprzedniej i nie należące do niej. Zbudujmy pierwszą sieć warstwową i wszystko powinno się wyjaśnić.



Zaczynamy zawsze od źródła, patrząc na sieć daną w zadaniu, szukamy wszystkich wierzchołków bezpośrednio osiągalnych z danego wierzchołka. Sam wierzchołek **s** stanowi warstwę zerową. Wypełnijmy pierwszą warstwę – z wierzchołka **s**, bezpośrednio osiągalne są wierzchołki **2**, **t**, **5** i **4** (będziemy zawsze przeglądali graf od góry do dołu). Wypisujemy je w jednej kolumnie, łączymy odpowiednimi łukami i nadajemy im etykiety zgodnie z istniejącymi połączeniami w sieci danej w zadaniu. Jednym z wierzchołków w utworzonej właśnie pierwszej warstwie jest ujście **t** – w związku z tym kończymy budowę sieci warstwowej (nie tworzymy kolejnej warstwy), bo znaleźliśmy najkrótszą ścieżkę prowadzącą ze źródła do ujścia

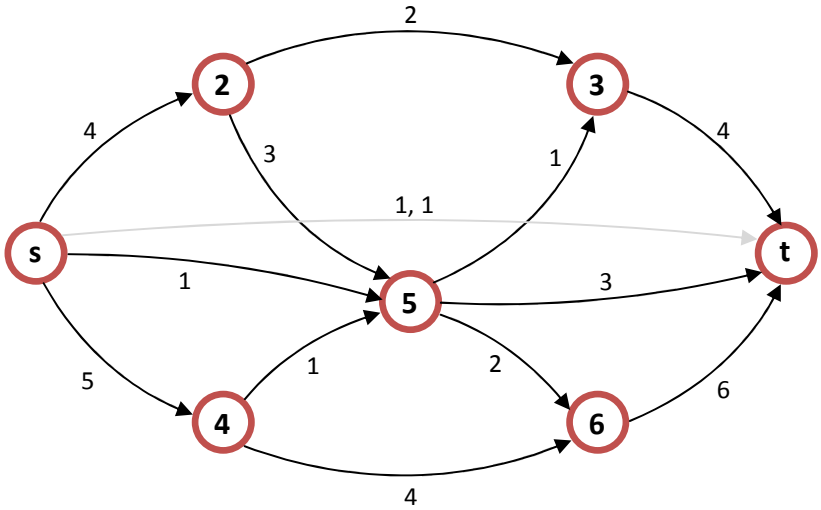
Przechodzimy do nasycania ścieżek użytecznych. Co to takiego? Jak pamiętamy, naszym zadaniem jest skonstruowanie maksymalnego dopuszczalnego przepływu ze źródła do ujścia w naszej sieci, korzystając z możliwych połączeń i mając na uwadze ich ograniczenia co do pojemności. W aktualnej sieci warstwowej szukamy wszystkich możliwych ścieżek prowadzących z **s** do **t**. Mamy tylko jedną taką ścieżkę, jest to przypadek trywialny bezpośredniego połączenia w postaci łuku s → t. Ma on etykietę „1”, to znaczy, że można tą ścieżką przesłać jedną jednostkę medium. O to nam właśnie chodzi, przyjmujemy tę ścieżkę jako część rozwiązania. Zaznaczamy ten fakt poprzez wypisanie wierzchołków wchodzących w ścieżkę powiększającą i wartość o jaką powiększyła nam przepływ. Dla naszego przypadku, zapis jest następujący: $st: \Delta = 1$.

Na naszej sieci warstwowej zaznaczamy nasyconą przez nas ścieżkę wraz z odpowiednimi wartościami etykiet. Konwencja przyjmowana w niniejszym kompendium jest następująca – nowe wartości etykiet dopisujemy po przecinku po prawej stronie od wartości poprzednich, przy czym jeśli dokonujemy kolejnych już zmian danej etykiety, to nowe wartości dodajemy po znaku plusa.



Zaktualizujmy zatem etykiety na pierwszej sieci warstwowej, dla lepszego zobrazowania sytuacji, będziemy zaznaczać odpowiednie ścieżki powiększające kolorami. Puściliśmy ścieżką s → t przepływ o wartości 1, zatem do jego etykiety dopisujemy po przecinku (bo jeszcze nic wcześniej tam nie dopisaliśmy) wartość 1. Pamiętajmy o zapisaniu obok sieci warstwowej wszystkich przepływów i wartości przez nie dodanych, w naszym przypadku mamy tylko $st: \Delta = 1$.

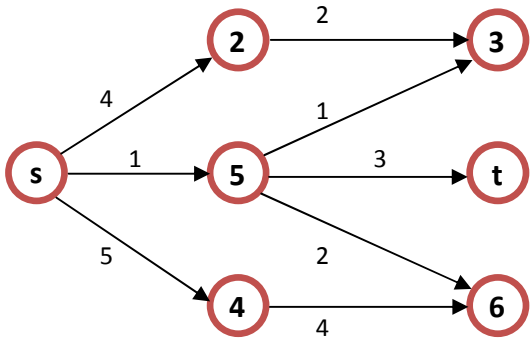
Musimy jeszcze w analogiczny sposób zaktualizować etykiety na oryginalnej sieci danej w zadaniu, aby przy budowie sieci warstwowych nie analizować nasyconych ścieżek (których nie możemy już używać, bo ich pojemność jest w całości wykorzystana).



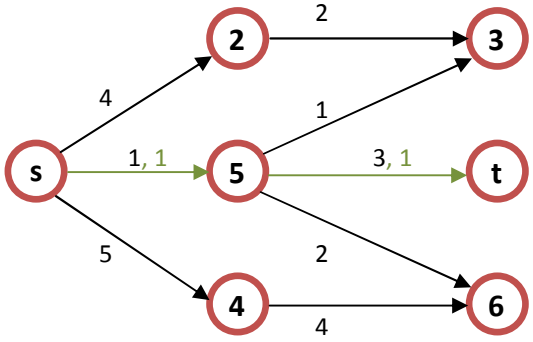
Ścieżka s → t ma pojemność 1 i przesłaliśmy nią 1 jednostkę medium, jest zatem nasycona (będziemy takie ścieżki zaznaczać szarym kolorem).

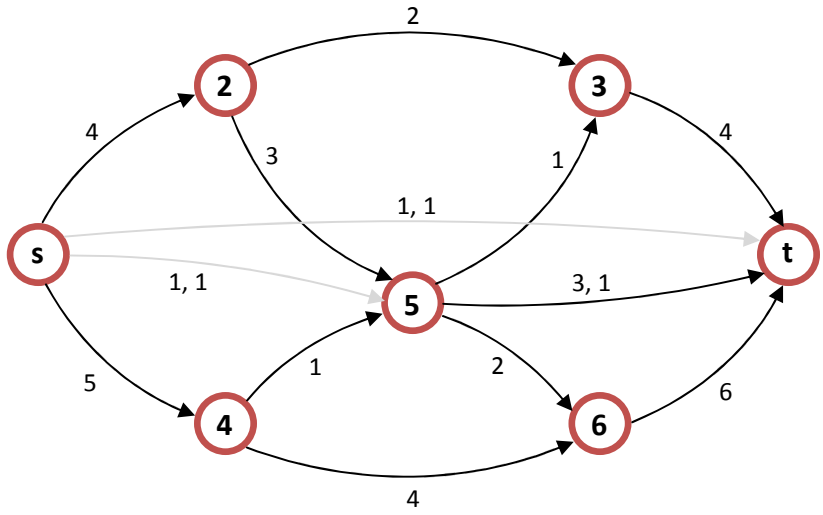
Skonstruujmy następną sieć warstwową. Zawsze zaczynamy od źródła s i tworzymy kolejne warstwy od niego w stronę ujścia. Patrzymy na zaktualizowaną sieć daną w zadaniu.

Z wierzchołka **s** bezpośrednio osiągamy wierzchołki **2**, **5** i **4**. Umieszczamy je zatem w pierwszej warstwie i łączymy odpowiednimi łukami z wierzchołkiem **s**, pamiętając o nadaniu im odpowiednich etykiet. Przeglądamy pierwsza warstwę od góry do dołu. Wierzchołek nr 2 jest połączony bezpośrednio z wierzchołkiem nr 3 i 5. W drugiej warstwie umieszczamy wierzchołek **3**, ale nie **5**, bo należy on już do tej samej warstwy, co rozpatrywany wierzchołek **2**. Łuk 2 → 5 jeszcze nam się przyda, ale z uwagi na tę regułę, nie dodajemy go do sieci warstwowej. Dalej, z wierzchołka nr 5 osiągalne są **3**, **t** i **6**. Dodajemy je do drugiej warstwy (wierzchołek **3** już tam jest) i łączymy łukami o odpowiednich etykietach. Obecność ujścia w tej warstwie mówi nam, że nie będziemy już dodawać nowych warstw do tej sieci, ale musimy rozpatrzyć jeszcze ostatni wierzchołek zanim przerwiemy budowę sieci warstwowej na tym etapie. Wierzchołek nr 4 prowadzi do wierzchołków nr 5 i 6. Sytuacja jak poprzednio, **5** należy do tej samej warstwy co **4** więc pomijamy te połączenie. Dodajemy za to połączenie z **6** i z uwagi na obecność **t**, kończymy budowę tej sieci warstwowej.



Poszukajmy ścieżek łączących źródło z ujściem. Istnieje tylko jedna, mianowicie s → 5 → t. Ile medium można nią przesłać? Pierwsze połączenie s → 5 pozwala na maksymalne przesłanie jednej jednostki, 5 → t na przesłanie trzech. W takim razie, wartość nasycenia tej ścieżki to najmniejsza z tych liczb, czyli $s5t\colon \Delta = 1$ (zapisujemy to obok sieci warstwowej). Zaktualizowana sieć warstwowa i sieć główna znajduje się poniżej (kolorowe strzałki występują tu tylko dla celów dydaktycznych, jednak samo zaznaczanie zmian w przepływie poprzez aktualizację etykiet jest niezbędne w rozwiązaniu tego zadania!).

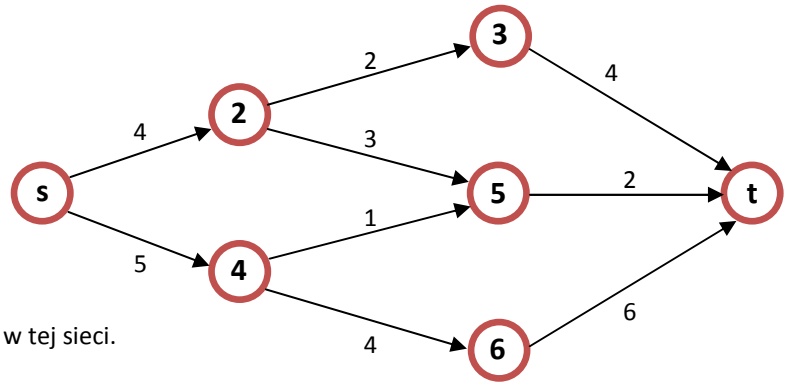




Pozbyliśmy się już dwóch łuków w głównej sieci z uwagi na ich nasycenie (zaznaczono powyżej szarym kolorem).

Zbudujmy zatem kolejną sieć warstwową. Wierzchołek **s** umieszczamy oczywiście w warstwie zerowej, można z niego osiągnąć wierzchołki **2** i **4**, zatem one wraz z odpowiednimi etykietami na łukach, lądują w warstwie pierwszej. Dalej z wierzchołka **2** osiągamy **3** i **5** (tym razem **5** umieszczamy w sieci warstwowej, bo występuje w innej warstwie niż **2**). Tak samo z **4** osiągamy wierzchołki **5** i **6**. Pamiętamy o połączeniu wierzchołków odpowiednimi łukami i nadanie im etykiet. Pierwsza warstwa zrobiona, przechodzimy do warstwy drugiej. Znajdują się tam wierzchołki **3**, **5** i **6**. Z **3** i **6** jedyne łuki biegną do **t**, dodajemy więc ujście do trzeciej warstwy i łączymy je z tymi wierzchołkami. Przyjrzyjmy się wierzchołkowi nr 5. Po pierwsze, można z niego dojść do 3 i 6

– dlaczego nie wprowadziliśmy takich połączeń w sieci warstwowej powinno już być jasne – te wierzchołki są w tej samej warstwie co nasz wierzchołek. Po drugie uważajmy na łuk 5 → t, ma on pojemność 3, ale wykorzystaliśmy już jedną jednostkę, więc możemy nim przesłać maksymalnie dwie jednostki i taką właśnie etykietę musimy mu nadać w naszej sieci warstwowej. Powstała sieć warstwowa znajduje się obok.

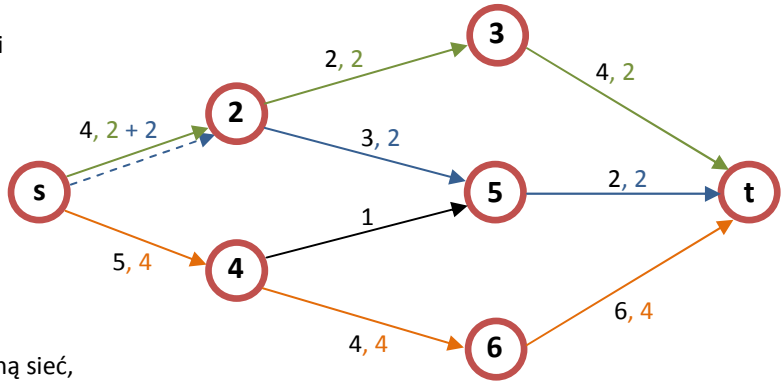


Postarajmy się teraz nasycić ścieżki użyteczne występujące w tej sieci.

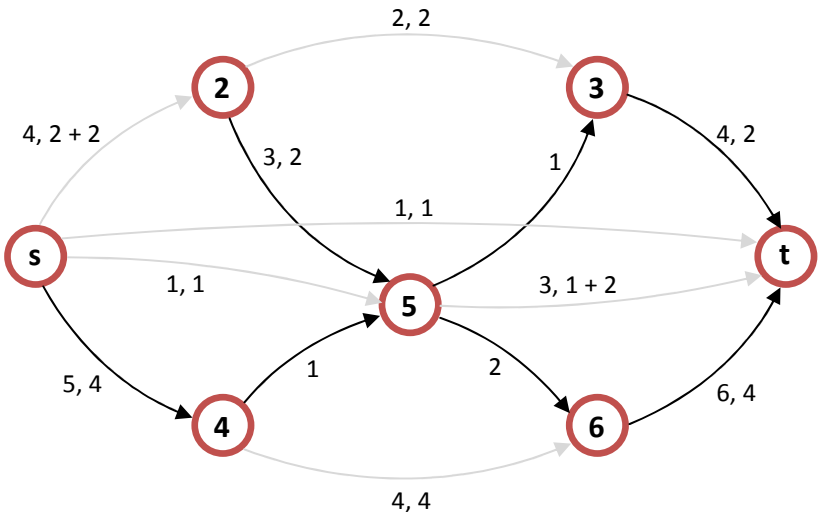
Przeglądając ją od góry, widzimy, że ścieżką s → 2 → 3 → t możemy przesłać dwie jednostki medium, wykorzystując dwie z czterech istniejących w łuku s → 2, obydwie istniejące w łuku 2 → 3 oraz dwie z czterech w łuku 3 → t. Zaznaczamy to odpowiednio po przecinkach na naszej sieci warstwowej. Dalej, ścieżką s → 2 → 5 → t możemy przesłać dwie jednostki, ponieważ w łuku s → 2 pozostały nam dwie z czterech początkowych, w 2 → 5 mamy wolne 3 jednostki, ale znów dwie jednostki w łuku 5 → t. Dopisujemy wykorzystane jednostki na odpowiednich łukach pamiętając, że zgodnie z przyjętą konwencją, jeśli już coś z nich wykorzystaliśmy, to kolejne liczby dopisujemy po plusie. Ostatnią ścieżką w tej sieci jest s → 4 → 6 → t, o przepustowości czterech jednostek (czyli najmniejszej z istniejących w ścieżce). Z łuków s → 4, 4 → 6 oraz 6 → t zabieramy po cztery jednostki, oznaczamy ten fakt na ich etykietach.

Trzy znalezione ścieżki powiększające zapisujemy obok sieci warstwowej tak jak poprzednio:

$s23t$: $\Delta = 2$
$s25t$: $\Delta = 2$
$s46t$: $\Delta = 4$

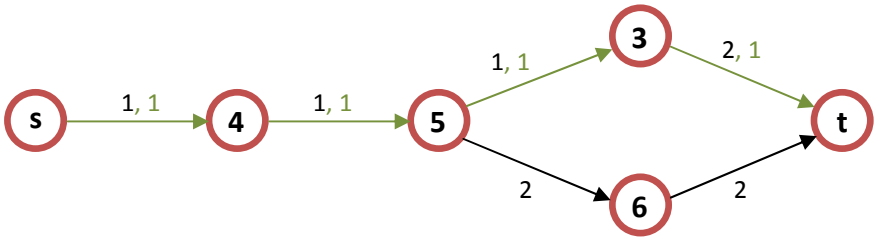


Pamiętajmy, że po nasyceniu łuku, znika on z sieci (tak głównej jak i warstwowej -- pomocniczej), staje się bezużyteczny do dalszych rozważań. Obok sieć warstwowa po nasyceniu ścieżek. Na jej podstawie aktualizujemy główną sieć, widoczną poniżej. Zauważmy, że mamy coraz mniej łuków do wykorzystania w dalszych etapach rozwiązania zadania, jesteśmy zatem coraz bliżej nasycenia wszystkich użytecznych ścieżek.

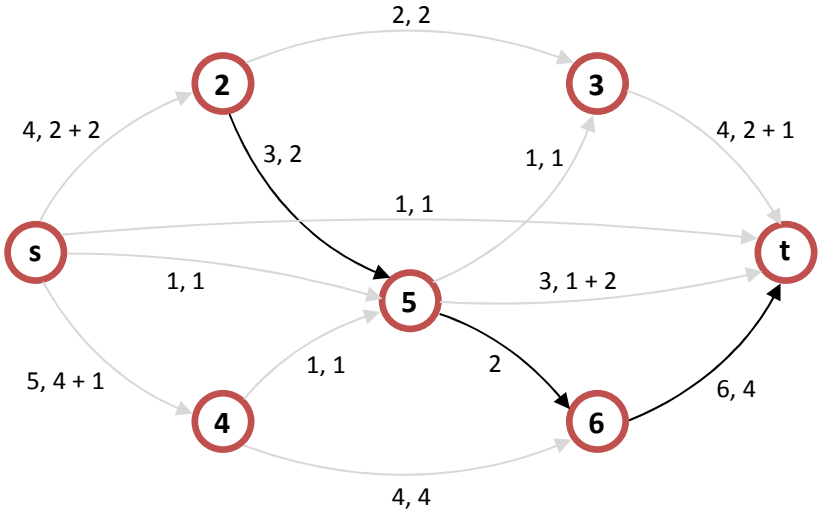


Przejdźmy do budowy kolejnej sieci warstwowej.

W warstwie zerowej umieszczamy wierzchołek **s**, w pierwszej warstwie wierzchołek **4**, w drugiej warstwie wierzchołek **5**, którym można dojść do wierzchołków **3** i **6** stanowiących trzecią warstwę i na końcu oczywiście w warstwie czwartej, wierzchołek **t**. Pamiętajmy, że etykiety odpowiednich łuków muszą uwzględniać nie tylko przepustowość kanału, ale też to, co w nim już płynie w wyniku naszych dotychczasowych działań. Sieć warstwowa powstała z tych wierzchołków wraz z zaznaczonym przepływem (już bez szczegółowych objaśnień) znajduje się poniżej, pamiętajmy o zapisaniu ścieżki powiększającej: $s453t$: $\Delta = 1$



Aktualizujemy główną sieć:



Przejdźmy do konstrukcji następnej sieci warstwowej. Z wierzchołka **s**… nie można nigdzie przejść. Ostatnia sieć warstwowa jest zatem postaci przedstawionej obok – składa się tylko z wierzchołka **s**, ze względu na brak dalszych połączeń (mimo wszystko należy ją narysować i to napisać).

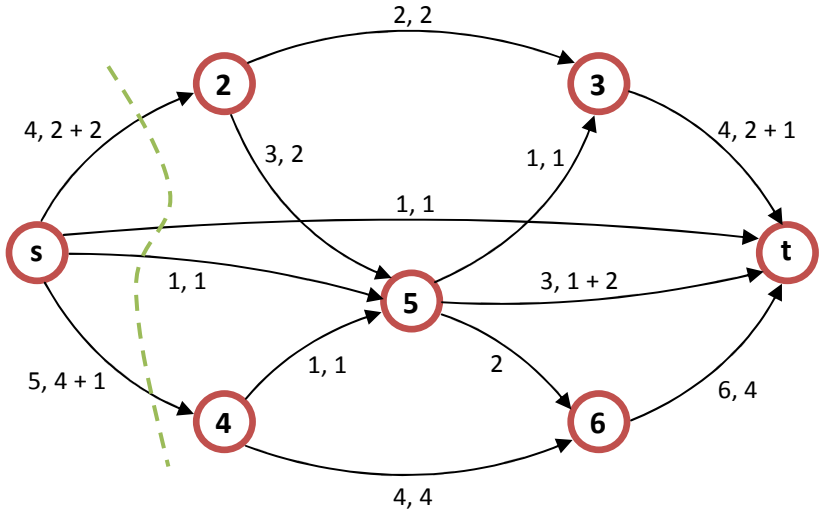
Znaleźliśmy zatem maksymalny przepływ w tej sieci, mógł on wyglądać trochę inaczej gdybyśmy wybrali inne ścieżki do nasycenia, ale puszczając siecią za każdym razem maksymalną liczbę jednostek, optymalną wartość przepływu otrzymalibyśmy taką samą.

W praktyce, tworząc sieci warstwowe, nawet jeśli z wierzchołka nic dalej nie wychodzi i nie ma w związku z tym szans dojścia przez niego do ujścia, umieszczamy go za każdym razem w odpowiedniej dla niego warstwie. Ważne jest zapisywanie przyrostów przepływu dokonanych za pomocą nasyconych ścieżek powiększających.

Znaleziony przepływ maksymalny to suma przepływów cząstkowych uzyskanych w kolejnych sieciach warstwowych. W naszym przykładzie jest on równy 1 + 1 + 2 + 2 + 4 + 1 = 11.

Skoro już się tak natrudziliśmy, warto znaleźć w naszej sieci tak zwane „wąskie gardło”. //co to jest dokładnie?

Robimy to w sposób następujący: przechodząc sieć od źródła **s** sposobem *w głąb*, jeśli natrafimy na łuk nasycony – odcinamy go i cofamy się do poprzedniego węzła. Jeśli łuk jest nasycony, idziemy dalej. Tym sposobem odcinamy łuki w naszej sieci i uzyskujemy ostatecznie poniższe rozwiązanie zadania.



SZEREGOWANIE ZADAŃ

Teoria

PRZYKŁAD

Rozwiąż następującą instancję problemu szeregowania $n = 4$ podzielnych zadań na $m = 3$ identycznych procesorach równoległych, o momentach gotowości $\bar{r} = [0, 1, 1, 2]$, czasach przetwarzania $\bar{p} = [3, 2, 4, 4]$ oraz liniach krytycznych $\bar{d} = [3, 3, 5, 4]$ dla kryterium największego opóźnienia ($P|pmtn, r_j|L_{max}$).

Zadanie polega na wyznaczeniu opóźnienia L_{max} z jakim maksymalnie spóźni się zadanie (lub więcej zadań) względem swojej linii krytycznej. Musimy uszeregować dane procesy do wykonania na danej liczbie procesorów, biorąc pod uwagę ich czas gotowości (czas, w którym pojawiają się w systemie), czas przetwarzania i linię krytyczną, czyli chwilę w której zadanie powinno być już zakończone. Ujemna wartość L_{max} oznacza, że wszystkie zadania zakończyły się przed swoimi liniami krytycznymi, zerowa oznacza zakończenie na czas, dodatnia to opóźnienie względem założonych linii krytycznych. L_{max} dotyczy przynajmniej jednego zadania, nie wszystkich zadań jako całości. Np. $L_{max} = 3$ oznacza, że przynajmniej jedno zadanie zostało zakończone trzy jednostki czasowe po jego linii krytycznej i te opóźnienie jest największe ze wszystkich (inne zadania też mogły się spóźnić, ale maksymalnie o 3 jednostki).

Rozwiązanie zadania składa się z kilku etapów, początkowo wykonujemy obliczenia, później korzystamy z naszej wiedzy na temat przepływów w sieciach aby odpowiednio uszeregować zadania na dostępnych procesorach. Obliczenia dokonujemy na zbiorach liczb, dlatego też, będziemy pomijać identyczne wielkości nic nie wnoszące do rozwiązania.

1. Obliczamy wartość $L_{max} \geq max\{r_j + p_j - d_j\}$

$$L_{max} \geq max\begin{Bmatrix} 0+3-3 \\ 1+2-3 \\ 1+4-5 \\ 2+4-4 \end{Bmatrix} \rightarrow L_{max} \geq max\begin{Bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{Bmatrix} \rightarrow L_{max} \geq 2$$

Odpowiednio sumujemy wszystkie wartości r_j i p_j dane w zadaniu, odejmujemy od nich odpowiednie d_j i największa z tych liczb daje nam ograniczenie na L_{max} (w obliczeniach nic się nie powtórzyło, dlatego nic nie pominęliśmy).

2. Obliczamy przedziały do których należy $L_{max} = \{r_i - d_j\}$

$$r_1 - d_j = \begin{Bmatrix} 0-3 \\ 0-5 \\ 0-4 \end{Bmatrix} = \begin{Bmatrix} -3 \\ -5 \\ -4 \end{Bmatrix}$$

$$r_2 - d_j = r_3 - d_j = \begin{Bmatrix} 1-3 \\ 1-5 \\ 1-4 \end{Bmatrix} = \begin{Bmatrix} -2 \\ -4 \\ -3 \end{Bmatrix}$$

$$r_4 - d_j = \begin{Bmatrix} 2-3 \\ 2-5 \\ 2-4 \end{Bmatrix} = \begin{Bmatrix} -1 \\ -3 \\ -2 \end{Bmatrix}$$

Wartość $d_1 = d_2$, dlatego pominęliśmy powyżej podwójne obliczenia. Uzyskaliśmy tutaj zbiór wartości granicznych dla L_{max}, za ich pomocą musimy teraz wyznaczyć przedziały występowania L_{max}, biorąc wszystkie unikalne liczby z punktu 2. I tworząc z nich przedziały prawostronnie domknięte, jak poniżej:

$$L_{max} \in (-\infty, -5\rangle, (-5, -4\rangle, (-4, -3\rangle, (-3, -2\rangle, (-2, -1\rangle, (-1, +\infty)$$

Zauważmy, że w punkcie 1. również obliczyliśmy warunek dla L_{max} (≥ 2), właśnie uzyskaliśmy drugi warunek, musimy zatem wyznaczyć ich logiczną koniunkcję postaci:

$$L_{max} \in \langle 2, +\infty)$$

Wiemy już zatem jakie możliwe wartości przyjmuje L_{max}.

3. Przyjmujemy wartość testową $L_{max} = 2$. Wartość ta może być poprawna lub nie – przyjmujemy ją równą początkowi przedziału, jeśli okaże się nieodpowiednia do naszego zadania (nie uda się uszeregować zadań z takim opóźnieniem), musimy sprawdzić kolejne wartości.

Obliczamy wektor $\bar{d}' = \bar{d} + L_{max}$ czyli nowe linie krytyczne dla zadań, uwzględniające testową wartość opóźnienia.

$\bar{d}' = [3, 3, 5, 4] + 2 = [5, 5, 7, 6]$

Obliczamy teraz wektor $\bar{e} = \{r\} \cup \{d'\}$ którego wartości stanowi suma mnogościowa elementów wektorów $\bar{r}$ i $\bar{d}'$, czyli wszystkie chwile czasowe w których zadania pojawiają się w systemie lub musi zakończyć się ich przetwarzanie. Mamy zatem:

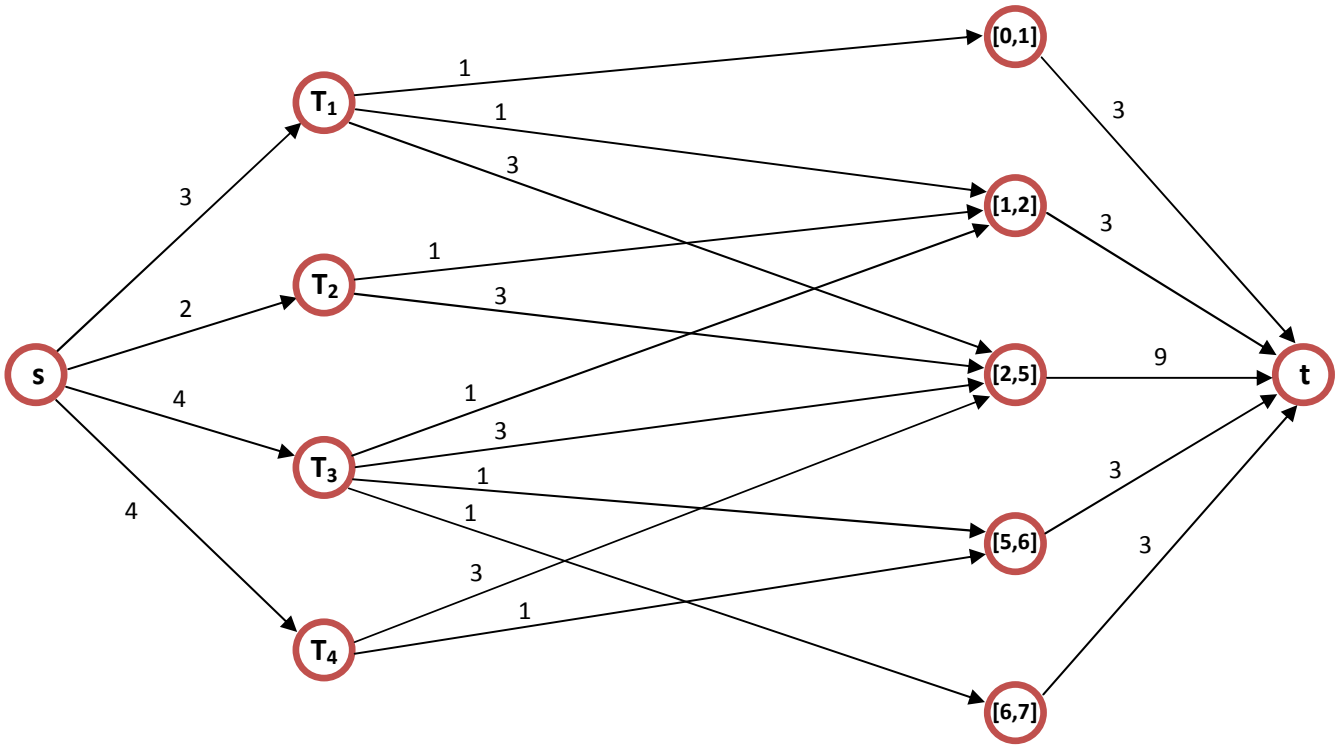
$\bar{e} = \{0, 1, 2\} \cup \{5, 6, 7\} = [0, 1, 2, 5, 6, 7]$

Przypisanie do wektora sumy zbiorów nie jest może poprawne formalnie, ale na pewno obrazowe – powstały zbiór musi być uporządkowany ze względów formalnych, stąd postać wektora $\bar{e}$.

4. Przystępujemy do wykreślenia wstępnej sieci przepływowej, za pomocą której uszeregujemy zadania na procesorach.

Kreślimy od lewej strony w kolumnach: źródło **s**, wierzchołki reprezentujące zadania $T_1, T_{2,\dots,} T_n$, wierzchołki reprezentujące przedziały chwil czasowych $[e_i, e_{i+1}]$ (teraz przydaje się wektor $\bar{e}$) i na końcu ujście sieci **t**.

Mając tak wykreślone wierzchołki, połączmy je łukami i nadajmy etykiety. Od **s** do wszystkich zadań T_n prowadzimy po jednym łuku o etykiecie równej liczbie p_n, czyli czasowi przetwarzania danego zadania. Dalej od wszystkich wierzchołków zadań do wierzchołków reprezentujących przedziały czasowe, prowadzimy łuki tylko wtedy, kiedy dane zadanie może być wykonywane w danym przedziale czasowym. W naszym przykładzie zadanie pierwsze ma czas gotowości $r_1 = 0$ a nową linię krytyczną (po wzięciu pod uwagę testowego opóźnienia!) $d'_1 = 5$, może zatem być wykonywane w przedziale czasu [0, 5] na który według naszych obliczeń (z wektora $\bar{e}$) składają się trzy przedziały [0, 1], [1, 2] oraz [2, 5]. Prowadzimy zatem po jednym łuku od T_1 do każdego z tych przedziałów, nadając mu etykietę równą szerokości przedziału (odpowiednio 1, 1 i 3). Ostatnie połączenia prowadzimy między wierzchołkami przedziałów czasowych a ujściem **t**, są to pojedyncze łuki o etykietach równych liczbie $m \cdot (e_{i+1} - e_i)$ czyli liczbie procesorów pomnożonej przez szerokość danego przedziału. Poniżej przygotowana przez nas sieć:



Uda nam się dokonać uszeregowania zadań, jeśli nasycimy wszystkie ścieżki s → T_j czyli zdołamy przydzielić tyle czasu ile potrzeba do przetworzenia wszystkich zadań mając na uwadze ograniczenia – czas przybycia zadań do systemu i ich linie krytyczne (uwzględniające wartość testową L_{max}).

W przykładzie będziemy nasycać ścieżki od góry do dołu, rozpatrując zadania po kolei. Nie ma jednego poprawnego rozwiązania tego zadania począwszy od tego miejsca. Uszeregowanie jest poprawne jeśli uda nam się osiągnąć jak najmniejsze L_{max}, które zadanie wykonuje się na którym procesorze, w jakiej kolejności je rozpatrujemy i kiedy się kończą lub zaczynają nie ma znaczenia, bo zadania są podzielne a procesory identyczne. Przejdźmy do puszczania zadań na określone procesory.

5. Przypisujemy zadania do wykonania na poszczególnych procesorach w danych przedziałach czasowych. Patrzymy wyłącznie na wykreśloną właśnie sieć. Rozpatrujmy kolejne zadania:

Zadanie T_1: $p_1 = 3$, musimy dostarczyć temu zadaniu 3 jednostki czasu, zaczynając od chwili $r_1 = 0$, najpóźniej kończąc w chwili $d'_1 = 5$. Możemy puścić te 3 jednostki poprzez:

$sT_1[0,1]t$: $\Delta = 1$
$sT_1[1,2]t$: $\Delta = 1$
$sT_1[2,5]t$: $\Delta = 1$

Tak też robimy, aktualizujemy etykietę łuku s → T_1 dodając przepływ 1 + 1 + 1, etykiety wszystkich łuków wychodzących z T_1, których użyliśmy właśnie do przesłania części naszego zadania oraz etykiety łuków między przedziałami czasowymi a ujściem **t**. Zarezerwowaliśmy sobie czas w tych przedziałach, należy poinformować o tym inne zadania. Powyższe działania zaznaczono na rysunku sieci kolorem zielonym.

Zadanie T_2: $p_2 = 2$ musimy dostarczyć temu zadaniu 2 jednostki czasu, zaczynając od chwili $r_2 = 1$, najpóźniej kończąc w chwili $d'_2 = 5$. Możemy puścić te 2 jednostki poprzez:

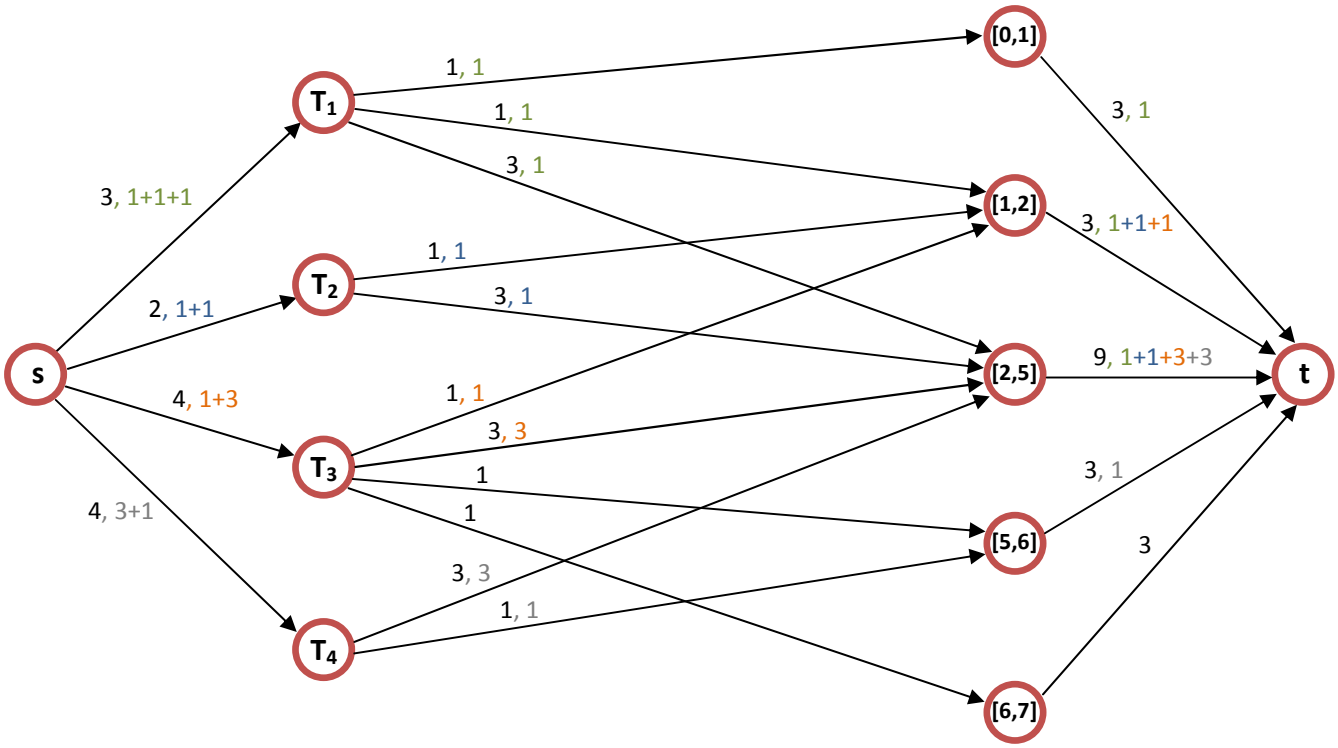
$sT_2[1,2]t$: $\Delta = 1$
$sT_2[2,5]t$: $\Delta = 1$

Aktualizujemy etykietę łuku s → T_2 dodając przepływ 1 + 1, etykiety wszystkich łuków wychodzących z T_2, których użyliśmy właśnie do przesłania części naszego zadania oraz etykiety łuków między przedziałami czasowymi a ujściem **t** (kolor niebieski).

Zadanie T_3: $p_3 = 4$ musimy dostarczyć temu zadaniu 4 jednostki czasu, zaczynając od chwili $r_3 = 1$, najpóźniej kończąc w chwili $d'_3 = 7$. Możemy puścić te 4 jednostki (kolor pomarańczowy) poprzez:

$sT_3[1,2]t$: $\Delta = 1$
$sT_3[2,5]t$: $\Delta = 3$

Zadanie T_4: $p_3 = 4$ musimy dostarczyć temu zadaniu 4 jednostki czasu, zaczynając od chwili $r_4 = 2$, najpóźniej kończąc w chwili $d'_4 = 6$. Możemy puścić dokładnie te 4 jednostki poprzez jedyne istniejące łuki (kolor szary). Zauważmy, że gdybyśmy nie przesunęli linii krytycznych, zadanie czwarte nie miałoby szansy wykonania, bo przybywając w chwili 2 i potrzebując 4 jednostki czasu na przetwarzanie, może się zakończyć najszybciej po czasie 2 + 4 = 6, jest to założenie niewykonalne – *infeasible*. Niezależnie od potrzeb innych zadań, nie możemy przetwarzać zadania równocześnie na wielu procesorach (możemy jednak ich przetwarzanie dzielić na części). Dla zadania czwartego brakuje nam dwóch jednostek czasu, stąd $L_{max} = 2$ okazuje się być poprawną wartością (i najmniejszą z możliwych). Ostatecznie sieć po przepuszczeniu przez nią zadań wygląda następująco:



6. Sporządźmy tabelę ilustrującą uszeregowanie zadań.

W pierwszej kolumnie tabeli mamy dostępne procesory, w ostatnim wierszu przedziały chwil a w odpowiednich komórkach zadania wykonywane na danym procesorze w danej chwili.

Spójrzmy na powyższą sieć zawierającą rozwiązanie problemu. Musimy przeanalizować wszystkie łuki łączące wierzchołki chwil czasowych z ujściem sieci **t**. Pierwszym łukiem [0, 1] → t jak widać (u nas po kolorze etykiet, bez kolorowania należy popatrzeć z których zadań łuki wchodzą do węzła danej chwili) przesyłane jest zadanie pierwsze, przez jedną jednostkę czasu (więcej nie można, jeśli przedział ma szerokość równą 1). Przydzielmy zatem w pierwszej jednostce czasu procesor P_1 zadaniu T_1. Z węzła chwili [0, 1] wychodzi tylko pierwsze zadanie (dochodzi do niego tylko pierwsze – nie sam łuk z pierwszego zadania, ale łuk którym zdecydowaliśmy się coś przesłać), kończymy zatem analizę pierwszego przedziału czasowego.

W drugim przedziale [1, 2] zdecydowaliśmy się przetwarzać trzy zadania, każde rzecz jasna przez jedną jednostkę czasu. Umieśćmy je zatem w tabeli – ich kolejność nie jest niczym narzucona, warto jednak układać je w miarę możliwości sensownie. Kontynuujmy zatem przetwarzanie zadania pierwszego na pierwszym procesorze, drugie przydzielmy do drugiego, trzecie do trzeciego.

Przejdźmy do wypełnienia przedziału [2, 5]. Ma on szerokość równą 3, czyli daje nam 9 komórek do wypełnienia zadaniami. Suma przydzielonych czasów procesora wynosi 1 + 1 + 3 + 3 = 8. Nie wypełniajmy tym razem kolejno zadań, spójrzmy na wszystkie naraz, żeby zachować czytelność tabeli. Zaczynając od najbardziej wymagających, czyli najtrudniejszych do ułożenia, zadania T_3 i T_4 wymagają trzech jednostek czasu, czyli zajmą wszystkie jednostki w przedziale [2, 5]. Trzy jednostki T_3 umieszczamy obok istniejącej już jednostki T_3 na procesorze P_3. Zadanie T_4 z jego trzema jednostkami przyporządkujmy do procesora P_1. Procesory P_1 i P_3 są już całkowicie zajęte w tym przedziale czasu. Zostało nam przyporządkowanie jednej jednostki do pierwszego i drugiego zadania. Żeby zminimalizować fragmentację, przydzielmy pierwszą wolną jednostkę do T_2, drugą do T_1 (fragmentacji nie dało się jak widać całkowicie uniknąć).

W przedziale [5, 6] wykonywane jest tylko zadanie czwarte, przydzielamy je do pierwszego procesora.

Tabela obok zawiera rozwiązanie zadania uszeregowania zadań. Puste komórki oznaczają przestój pracy procesorów. Nie jest to jedyne poprawne uszeregowanie, jednak bez wątpienia optymalne a o to chodzi w zadaniu.

P_3		T_3	T_3	T_3	T_3		
P_2		T_2	T_2	T_1			
P_1	T_1	T_1	T_4	T_4	T_4	T_4	
	[0, 1]	[1, 2]	[2, 3]	[3, 4]	[4, 5]	[5, 6]	[6, 7]

PROBLEM PLECAKOWY

Teoria

PRZYKŁAD

Stosując metodę alternatywną, rozwiąż następującą instancję problemu plecakowego: $\bar{w} = [5, 3, 2, 4, 3]$, $\bar{s} = [3, 4, 2, 6, 1]$, $b = 10$.

W metodzie alternatywnej, kolejne wiersze tabeli odpowiadają rozpatrywanym elementom, kolumny odpowiadają osiąganym wartościom elementów włożonych do plecaka, wartości komórek na przecięciu wierszy i kolumn odpowiadają natomiast rozmiarowi (kosztowi) umieszczonych w plecaku elementów, osiągających daną wartość.

Wektor $\bar{w}$ oznacza wartości kolejnych elementów. Wektor $\bar{s}$ oznacza ich rozmiary (koszty). Wartość b to pojemność plecaka.

Metoda rozwiązania problemu polega na wykreśleniu odpowiedniej tabeli i sukcesywnym wypełnianiu jej w kolejnych iteracjach dla kolejnych elementów. W przykładach pomijamy zerowy wiersz i zerową kolumnę – oznaczają one przypadki trywialne, kiedy odpowiednio: rozpatrujemy zero elementów do umieszczenia w plecaku, oraz osiągnęliśmy zerową wartość elementów w plecaku. Puste komórki oznaczają nieskończoność.

Kreślimy tabelę o liczbie wierszy równej liczbie elementów, czyli 5. Oznaczają one odpowiednio 1 – pierwszy element znajduje się w zbiorze rozpatrywanych elementów, 2 – pierwszy i drugi element znajduje się w zbiorze rozpatrywanych elementów, 3 – rozpatrujemy pierwszy, drugi i trzeci, itd. Ile potrzebujemy kolumn w naszej tabeli? Maksymalnie tyle, ile wynosi suma wartości wszystkich elementów danych w zadaniu. $\sum w_i = 17$, zatem kreślimy kolumny o wartościach od 1 do 17.

	1	2	3	4	5
$\bar{w}$	5	3	2	4	3
$\bar{s}$	3	4	2	6	1

Obok tabeli na której będziemy pracować, należy jeszcze umieścić zawartość wektorów danych w zadaniu wraz z numerami poszczególnych elementów, tak aby wartości elementów znajdowały się nad ich rozmiarami (tak samo jak w tabeli z rozwiązaniem, w nagłówku mamy wartości, poniżej ich, w komórkach – rozmiary). Nie jest to wymóg, tak samo jak nadawanie numerów elementom, ale wyraźnie przyśpiesza pracę i zmniejsza ryzyko pomyłek. Mamy zatem tabelę główną i pomocniczą.

Przechodzimy do pierwszej iteracji algorytmu alternatywnego, rozwiązującego problem plecakowy. Dla przejrzystości ilustracje będą dotyczyły tylko wierszy rozpatrywanych w danej iteracji.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												

Jesteśmy w pierwszym wierszu. W zbiorze rozpatrywanych elementów mamy zatem tylko element pierwszy. Element 1 po włożeniu do plecaka, pozwala nam osiągnąć wartość 5 przy rozmiarze 3. Wpisujemy zatem w piątej komórce pierwszego wiersza rozmiar 3. Mając tylko jeden element, nie możemy rozpatrzyć innych możliwości umieszczenia elementów w plecaku, przechodzimy zatem do kolejnej iteracji algorytmu, do wiersza drugiego, w którym do dyspozycji mamy element 1 i 2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4														

Jesteśmy w drugim wierszu. Jako możliwe rozwiązanie rozpatrujemy kombinacje elementów 1 i 2. Umieszczamy najpierw element 2 w plecaku – ma on wartość 3 i rozmiar 4, zatem w trzeciej komórce aktualnego wiersza wpisujemy 4.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4					7									

Teraz uzupełniamy nasz wiersz na podstawie wartości z poprzedniego wiersza. Patrząc na wszystkie wypełnione komórki w poprzednim wierszu zastanawiamy się, jak zmieni się wartość i rozmiar naszego plecaka, jeśli do kombinacji poprzednich elementów, dodamy aktualny element. Jesteśmy w wierszu drugim, uzupełniamy go zatem na podstawie pierwszego. Przeglądamy wszystkie wypełnione komórki od lewej do prawej.

Co mówi nam liczba 3 w komórce nr 5 pierwszego wiersza? Mając do dyspozycji tylko element pierwszy, osiągnięcie wartości 5 jest możliwe przy najmniejszym rozmiarze 3. Dodajmy zatem nasz nowy element do tej kombinacji i sprawdźmy czy poprawi nam to rozwiązanie. W komórce [*rozpatrywany element*, *wartość poprzednia* + *wartość rozpatrywanego elementu*] (biorąc pod uwagę element [1, 5] z poprzedniego wiersza) czyli [2, 5 + 3] wpisujemy *poprzedni rozmiar* + *rozmiar rozpatrywanego elementu*, czyli 3 + 4. Nie ma w poprzednim wierszu innych elementów, kończymy ten etap algorytmu i przechodzimy do ostatniego.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									

Przepisujemy wartości z poprzedniego wiersza do aktualnego, jeśli poniżej ich w naszym wierszu mamy pustą komórkę, czyli rozmiar równy nieskończoności.

3 < ∞, zatem w do komórki [2, 5] przepisujemy rozmiar z [1, 5]. Iteracja druga zakończona.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2															

W iteracji trzeciej rozpatrujemy elementy 1, 2 i 3 jako możliwe do umieszczenia w plecaku.

Umieszczamy nasz element nr 3, ma on wartość 2 i rozmiar 2. Wartość nie była dotąd nigdy osiągnięta, więc rozmiar 2 jej odpowiadający jest najlepszy z możliwych. Zatem w komórce [3, 2] (czyli [*numer elementu*, *jego wartość*]) wpisujemy 2 (*rozmiar elementu*).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2			3												

Przechodząc po wszystkich wartościach z poprzedniego wiersza, staramy się poprawić uzyskany wcześniej wynik dodając aktualny element do plecaka.

[2, 3] = 4, staramy się poprawić rozmiar w komórce [3, 3 + 2] na 4 + 2, niestety 6 jest gorszym wynikiem niż wcześniej uzyskany w komórce [2, 5] rozmiar 3, zatem przepisujemy poprzedni rozmiar (najlepszy do tej pory).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2			3		5										

[2, 5] = 3, spróbujmy poprawić [3, 5 + 2] na 3 + 2. Oczywiście 5 < ∞, zatem w komórce [3, 7] wpisujemy rozmiar 5.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2			3		5			9							

[2, 8] = 7, spróbujmy poprawić [3, 8 + 2] na 7 + 2. Oczywiście 9 < ∞, zatem w komórce [3, 10] wpisujemy rozmiar 9.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							

Nie ma już więcej wypełnionych komórek w poprzednim wierszu, zatem przechodzimy do ostatniego etapu tej iteracji – przepisujemy do pustych komórek naszego wiersza wartości z poprzedniego. Jak można zauważyć, liczby w tabeli będą układać się zawsze na kształt wykresu słupkowego.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6													

Iteracja czwarta, wprowadzamy czwarty element do plecaka. Za jego pomocą, wartość 4 osiągamy rozmiarem 6. Powyżej komórki [4, 4] nie znajduje się żaden rozmiar lepszy niż 6, dlatego też wpisujemy tam rozmiar naszego aktualnego elementu.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6		8											

Przechodzimy po kolejnych elementach z poprzedniego wiersza i staramy się poprawić poprzednie wartości dokładając nasz aktualny element.

W komórce [3, 2] mamy rozmiar 2, przechodzimy do komórki [4, 2 + 4] i ponieważ wcześniej nie osiągnęliśmy takiego rozmiaru, wpisujemy w niej rozmiar 2 + 6.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6		8	5										

[3, 3] = 4, przechodzimy do komórki [4, 3 + 4] i próbujemy poprawić rozmiar powyżej (5) na 4 + 6. Nic z tego, zatem przepisujemy rozmiar z [3, 7] do [4, 7] (możemy też zostawić komórkę pustą).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6		8	5		9								

[3, 5] = 3, przechodzimy do komórki [4, 5 + 4] i poprawiamy jej zawartość na 3 + 6.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6		8	5		9								

[3, 7] = 5, spróbujmy poprawić zawartość komórki [4, 7 + 4] na 5 + 6. Wygląda na to, że możemy tak zrobić, ale zwróćmy uwagę pojemność plecaka. Wynosi ona 10. Wpisanie do komórki [4, 11] rozmiaru 11 oznaczałoby, że osiągnęliśmy najlepszą wartość spośród dotychczasowych (11), ale kosztem przekroczenia dopuszczalnego rozmiaru (kosztu) elementów wchodzących do plecaka w danej kombinacji. Nie możemy tak zrobić, dlatego przechodzimy do następnych elementów poprzedniego wiersza i pozostawiamy rozpatrywaną komórkę pustą.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6		8	5		9								

[3, 8] = 7, dodając wartość aktualnego elementu lądujemy w kolumnie nr 12, czyli osiągamy największą dotychczas wartość elementów w plecaku. Niestety, rozmiar dotychczasowych elementów obecnych w plecaku razem z rozmiarem aktualnego elementu przekracza dopuszczalne maksimum (b = 10 < 7 + 6), nie możemy go do nich dołożyć.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4				6		8	5		9								

Sytuacja identyczna jak poprzednie, widać to już zresztą po rozmiarze w komórce [3, 10] = 9, który po zsumowaniu z rozmiarem naszego elementu znacząco przekroczy dopuszczalny limit.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4		2	4	6	3	8	5	7	9	9							

Przepisujemy rozmiary z poprzedniego wiersza, jeśli odpowiadające im komórki w wierszu aktualnym są puste.

Koniec czwartej iteracji.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4		2	4	6	3	8	5	7	9	9							
5			1														

Przechodzimy do piątego wiersza.

Dodajemy nowy element do rozpatrywanego zbioru. Jak widać z tabeli pomocniczej, za jego pomocą osiągamy wartość 3 rozmiarem 1. To pierwszy przypadek w tym przykładzie, żeby w kolumnie odpowiadającej wartości naszego elementu istniały już wcześniej jakieś rozmiary (dzieje się tak oczywiście zawsze, kiedy dodajemy kolejny element o wartości, która już wcześniej wystąpiła w innym elemencie). Te rozmiary jednak, są gorsze niż rozmiar naszego elementu, zatem poprawiamy je poprzez wpisanie rozmiaru 1 w kolumnie odpowiadającej wartości 3. Przystępujemy teraz do drugiego etapu, czyli przechodzenia po wszystkich elementach z poprzedniego wiersza.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4		2	4	6	3	8	5	7	9	9							
5			1														

W celu zaoszczędzenia miejsca, nie będziemy już rozpatrywać tego szczegółowo. Do tej pory można było zauważyć, że za każdym razem, kiedy rozpatrujemy jakąś komórkę z poprzedniego wiersza, nowy rozmiar wpisujemy w wierszu niżej (aktualnym) przesuwając się w lewo o tyle kolumn, ile wynosi wartość aktualnego elementu, a wpisując tam rozmiar równy sumie rozmiaru w rozpatrywanej komórce z poprzedniego wiersza oraz rozmiaru aktualnego elementu. Odpowiadające sobie pary komórek z poprzedniego i aktualnego wiersza zaznaczono na ilustracji naprzemiennymi kolorami. Dla przypomnienia, przechodząc po wszystkich elementach z poprzedniego wiersza czyli komórkach o współrzędnych [*poprzedni wiersz*, *rozpatrywana wartość*] o

zawartości *poprzedni rozmiar*, sprawdzamy czy rozmiar równy *poprzedni rozmiar + rozmiar aktualnego elementu* wpisany w komórce [*aktualny wiersz*, *rozpatrywana wartość + wartość aktualnego elementu*] poprawi nam rozmiar występujący powyżej w tej samej kolumnie. Jeśli wcześniej występowała nieskończoność (pusta komórka), to dowolny skończony rozmiar poprawi nam wynik dla danej wartości elementów w plecaku, o ile nie przekroczymy tym samym pojemności samego plecaka. Jeśli nie poprawimy rozmiaru z komórki powyżej (z poprzedniego wiersza), to przepisujemy ten powyższy rozmiar (możemy też nic nie robić, przepiszemy ten rozmiar w ostatnim etapie danej iteracji – w niniejszych przykładach przepisujemy od razu).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4		2	4	6	3	8	5	7	9	9							
5			1		3	5	5	4	9	6	8	10	10				

Po przejściu wszystkich komórek z poprzedniego wiersza, tabela tak, jak widać na ilustracji obok. Przejdźmy do ostatniego etapu ostatniej już iteracji.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4		2	4	6	3	8	5	7	9	9							
5		2	1	6	3	5	5	4	9	6	8	10	10				

Przepisujemy rozmiary z poprzedniego wiersza do wiersza aktualnego, jeśli tylko zostały nam w nim jakieś puste komórki.

Cały algorytm alternatywnego rozwiązania problemu plecakowego został niniejszym wykonany. Aby rozwiązać do końca zadanie, musimy ustalić które elementy znajdują się w najlepszym zestawie elementów. Robimy to następująco:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1					3												
2			4		3			7									
3		2	4		3		5	7		9							
4		2	4	6	3	8	5	7	9	9							
5		2	1	6	3	5	5	4	9	6	8	10	10				

W ostatnim wierszu szukamy rozmiaru położonego najbardziej na prawo (dającego nam największą wartość elementów). Jest to rozmiar 10 w kolumnie 13 (nie musi być wcale równy pojemności plecaka ani być największy w całym wierszu!).

Jeśli w wierszu powyżej jest jakiś rozmiar różny od rozpatrywanego, to rozpatrywany wiersz wskazuje nam element należący do rozwiązania. Od rozmiaru we wskazywanej komórce odejmujemy rozmiar właśnie wybranego elementu (odpowiadający wierszowi w którym znajduje się wskazywana komórka) i przechodzimy do wiersza powyżej a w nim szukamy skrajnie prawej komórki zawierającej wyliczony rozmiar. Dla niej powtarzamy operację.

Jeśli natomiast w wierszu powyżej jest rozmiar równy rozpatrywanemu, to przechodzimy do wiersza wyżej i rozpatrywany dotychczas element nie należy do rozwiązania. Powtarzamy operację dla komórki do której przeszliśmy (znajdującej się w tej samej kolumnie, tylko wiersz wyżej).

W naszym przykładzie rozwiązanie wygląda następująco: skrajnie prawą komórką w ostatnim wierszu jest [5, 13] o rozmiarze 10. Rozmiar nad nią jest różny od 10, zatem element reprezentowany przez wiersz piąty (czyli element nr 5) wchodzi do rozwiązania. Odejmujemy od aktualnego rozmiaru plecaka rozmiar elementu piątego, mamy 10 – 1 = 9. Idziemy wiersz wyżej, do wiersza czwartego i szukamy rozmiaru 9 występującego jak najbardziej na prawo. Jest to komórka [4, 10]. Nad nią niestety również znajduje się rozmiar 9, czyli element z aktualnego wiersza nie wchodzi do rozwiązania, przechodzimy od razu do wiersza wyżej. W wierszu trzecim, rozpatrujemy zatem komórkę [3, 10]. Nad nią nie występuje rozmiar równy jej rozmiarowi, czyli dany element wchodzi do rozwiązania – odejmujemy od aktualnego rozmiaru plecaka rozmiar trzeciego elementu i mamy 9 – 2 = 7. Idziemy do wiersza wyżej i szukamy w nim skrajnie prawego rozmiaru równego 7. Znajduje się on w komórce [2, 8], nad nią nie ma rozmiaru 7, więc element drugi wchodzi do rozwiązania. Odejmujemy od aktualnego rozmiaru plecaka rozmiar drugiego elementu, mamy 7 – 4 = 3. Przechodzimy wiersz wyżej i szukamy skrajnie prawego rozmiaru równego 3. Znajduje się on w komórce [1, 5], nad nią nie ma rozmiaru równego 3, więc ten element (nr 1) też wchodzi do rozwiązania. Po odjęciu rozmiaru pierwszego elementu od rozmiaru plecaka, mamy 3 – 3 = 0, czyli wszystko w porządku. Oznacza to, że wyjmując z plecaka po kolei wszystkie elementy stanowiące najlepszy zestaw do zapakowania, całkowicie go opróżniliśmy.

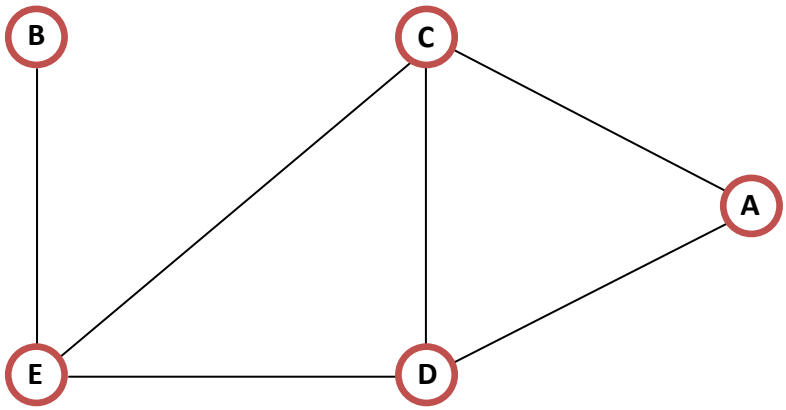
Rozwiązanie zadania daje nam informację, że do plecaka wchodzą elementy 1, 2, 3 i 5.

KOLOROWANIE GRAFÓW

Teoria (ograniczenia + coś o greedym)

PRZYKŁAD

Znajdź $X(G)$ czyli liczbę chromatyczną poniższego grafu.



Liczba chromatyczna jest to najmniejsza liczba kolorów jaką można pokolorować wszystkie wierzchołki danego grafu.

Posłużymy się dedykowanym algorytmem dokładnym kolorowania wierzchołkowego grafu, tak zwanym algorytmem Browna.

Podobnie jak w innych przykładach, przykładowa tabela będzie się rozrastała w zależności od potrzeb. Pierwszym krokiem jest pokolorowanie grafu metodą zachłanną według kolejności etykiet wierzchołków. Metoda zachłanna polega na nadawaniu każdemu kolejnemu wierzchołkowi najniższego możliwego koloru.

A	B	C	D	E
1	1	2	3	4

Zaczynamy od wierzchołka **A**. Żaden z sąsiadujących z nim wierzchołków nie ma ustalonego koloru, możemy więc pokolorować go kolorem 1. Wierzchołek **B** również nie ma żadnego pokolorowanego sąsiada, otrzymuje również kolor 1. Wierzchołek **C** ma sąsiada **A** pokolorowanego kolorem 1, zatem najmniejszy możliwy dla niego to kolor 2, który mu przyznajemy. Wierzchołek **D** będąc w sąsiedztwie z **A** oraz **C**, które mają już kolory 1 i 2, musi otrzymać kolor 3. Najbardziej pechowy okazuje się być wierzchołek **E**, którego wszyscy sąsiedzi (tak, tak – personifikujemy wierzchołki ze względów dydaktycznych) mają już ustalone kolory na 1, 2 i 3. Nie mamy wyjścia i przyporządkowujemy mu kolor 4.

Niniejszym zakończyliśmy wstępny etap algorytmu – graf został pokolorowany metodą zachłanną.

Jak widać, udało się nam pokolorować graf czterema kolorami. Intuicja rasowego informatyka powinna już nam podpowiadać, że nie jest to kolorowanie optymalne.

Przejdźmy zatem do właściwego działania algorytmu. Wyróżniamy w nim dwie fazy, pierwsza, którą teraz zastosujemy nazywa się *backtracking*, cofamy się w niej przechodząc wierzchołki od prawej strony tabeli do lewej, w sposób następujący:

Spróbuj pokolorować wierzchołek kolorem mniejszym niż liczba użytych dotychczas kolorów i takim, który nie był już wcześniej użyty do pokolorowania tego wierzchołka.

A	B	C	D	E		
1	1	2	3	4	←	<4

Zapiszmy najpierw szukany maksymalny kolor użyty w kolorowaniu (obok głównej tabeli) i kierunek działania (strzałka).

Przechodząc tabelę od strony prawej do lewej, spróbujmy według powyższej reguły pokolorować wierzchołek **E** kolorem mniejszym niż aktualnie najlepszy i takim, który nie był nigdy użyty do jego pokolorowania (w praktyce wybieramy najniższy możliwy, aby zachować pewną konsekwencję rozwiązywania przykładu i uniknąć pomyłek). Chcemy zatem pokolorować go kolorem mniejszym niż 4 i różnym od 4, sprawdźmy je po kolei: 1 odpada, bo jest nim pokolorowany **B**, 2 odpada, bo jest nim pokolorowany **C**, 3 odpada, bo jest nim pokolorowany **D**. Nie udało się, w związku z tym:

A	B	C	D	E		
1	1	2	3	X	←	<4

Jeśli się nie uda, usuń kolor tego wierzchołka i przejdź do poprzedniego w tabeli, spróbuj pokolorować nowy wierzchołek w taki sam sposób.

Usuwamy kolor wierzchołka **E**, od tej pory traktujemy go tak, jakby nie miał przydzielonego koloru (nie będzie zatem przeszkadzał swoim sąsiadom w przydzielaniu sobie kolorów).

ZAGADNIENIA TEORETYCZNE

ZAGADNIENIE 1

Teoria 1

ZAGADNIENIE 2

Teoria 2