Cz.1

piątek, 28 maja 2021 12:50

Teoria Grafów - projekt zaliczeniowy

Zadania dla: Krzysztof Łazarz

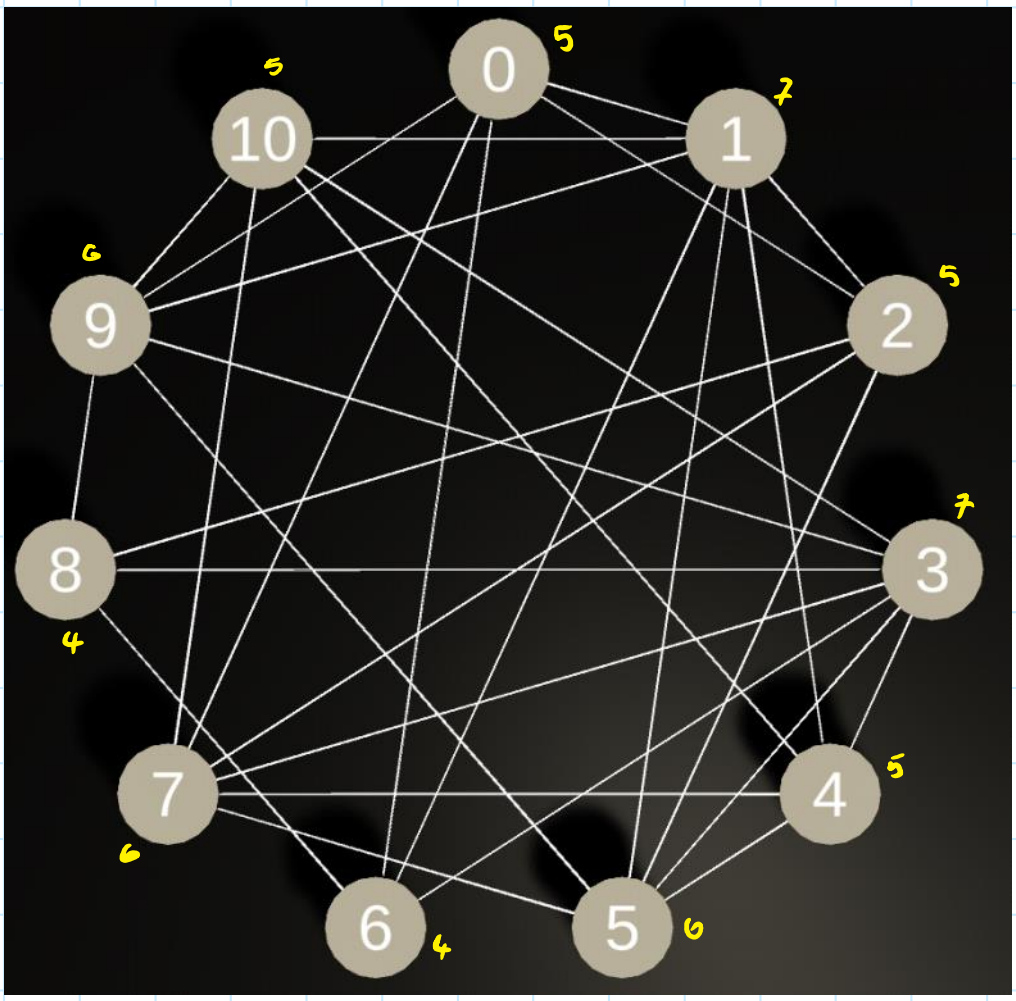
Część analityczna

W załączniku, w pliku Krzysztof_Łazarz.json znajduje się lista sąsiedztwa dla grafu do przeanalizowania. Zadania w części analitycznej (1-8) mają zostać wykonane w oparciu o ten właśnie graf. Zadania mogą być rozwiązane na kartce i zeskanowane lub wykonane w dowolnym programie, np. OneNote. Proszę o wyniki w formie pliku pdf.

Zadanie 1 (1pkt)

Wykonaj szkic grafu.

(Szkic wykonany przy pomocy autorskiej aplikacji)



0: [2, 1, 7, 9, 6],
1: [6, 9, 0, 2, 5, 4, 10],
2: [0, 5, 8, 1, 7],
3: [4, 7, 10, 9, 6, 5, 8],
4: [3, 5, 7, 10, 1],
5: [7, 2, 4, 9, 1, 3],
6: [1, 8, 3, 0],
7: [3, 5, 10, 0, 2, 4],
8: [9, 2, 6, 3],
9: [8, 1, 3, 5, 10, 0],
10: [3, 7, 9, 4, 1],

Graf jest nieskierowany

Zadanie 2 (1pkt)

Opisz graf w formie macierzy incydencji.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1		numery wierzchołków																														
2	numer krawędzi		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
3		0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4		1	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5		2	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6		3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
7		4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
8		5	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0
9		6	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
10		7	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0
11		8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0
12		9	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1
13		10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1	

Dodaj więcej wierszy (1000)u dołu.

Zadanie 3 (3pkt)

Czy ten graf jest hamiltonowski/pół-hamiltonowski? Jeśli tak to podaj ścieżkę/cykl Hamiltona.

Graf jest hamiltonowski
Przykładowy cykl hamiltona:
[0, 2, 5, 7, 3, 4, 10, 9, 8, 6, 1, 0]

Zadanie 4 (3pkt)

Czy ten graf jest eulerowski/pół-eulerowski? Jeśli tak to podaj ścieżkę/cykl Eulera.

Nie jest eulerowski ani półeulerowski, ponieważ mamy 6 wierzchołków (0,1,2,3,4,10) nieparzystego stopnia, a Euler dopuszcza ilość wynoszącą 0 lub 2 żeby stworzyć cykl lub ścieżkę eulera

Zadanie 5 (2pkt)

Pokoloruj graf wierzchołkowo oraz krawędziowo.

Zadanie 6 (1pkt)

Podaj liczbę chromatyczną oraz indeks chromatyczny dla grafu.

Liczba chromatyczna = 4

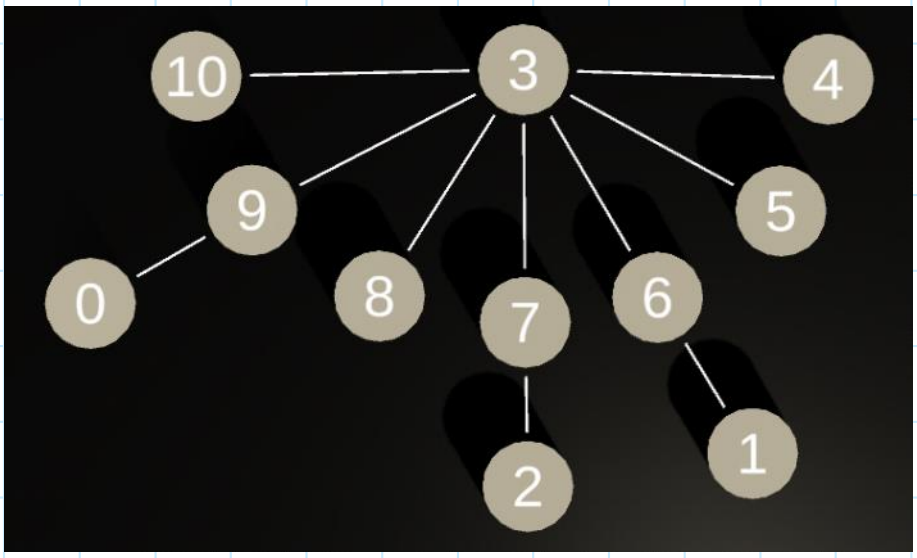
Wierzchołek	kolor
0	0
1	3
2	2
3	3
4	2
5	0
6	1
7	1
8	0
9	1
10	0

Indeks chromatyczny = 7

Połączenie	kolor
1-0	0
1-2	1
1-4	2
1-5	3
1-6	4
1-9	5
1-10	6
9-10	0
8-9	1
7-10	1
6-8	0
5-9	2
5-7	0
4-10	3
4-7	4
4-5	1
3-10	2
3-9	3
3-8	4
3-7	5
3-6	1
3-5	6
3-4	0
2-8	2
2-7	3
2-5	4
0-9	4
0-7	2
0-6	3
0-2	5

Zadanie 7 (1pkt)

Wyznacz minimalne drzewo rozpinające dla analizowanego grafu.



Graf został tak ułożony aby zminimalizować średni czas przejścia, lub inaczej żeby zminimalizować sumę kosztów przejścia z każdego wierzchołka do każdego innego

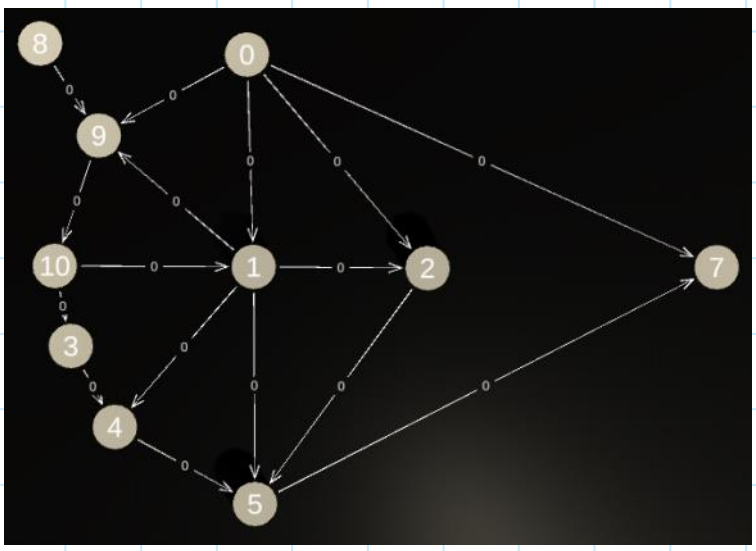
$$\sum_{a \in V} \sum_{b \in V} distance(a, b)$$

Zadanie 8 (2pkt)

Czy rysunek tego grafu jest planarny? Jeśli nie, to czy da się go przedstawić jako planarny? Jeśli tak, to ile ścian można w nim wyznaczyć? Proszę to wykazać na rysunku

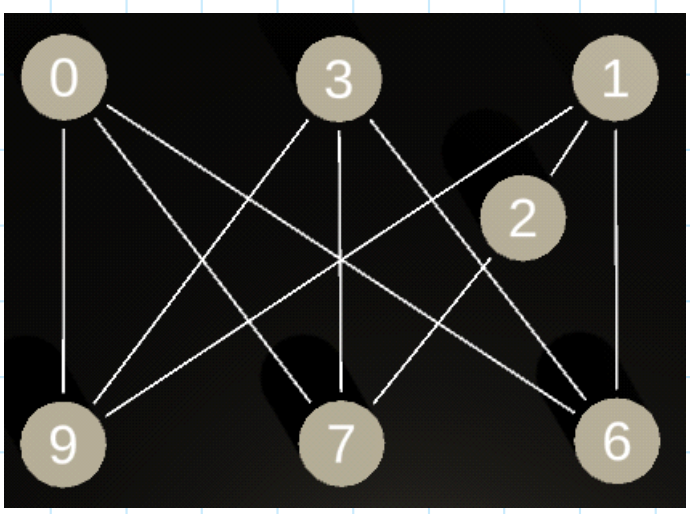
Dowód intuicyjny:
Zacząłem rysowanie wejściowego grafu:
0: [2, 1, 7, 9, 6],
1: [6, 9, 0, 2, 5, 4, 10],
2: [0, 5, 8, 1, 7],
3: [4, 7, 10, 9, 6, 5, 8],
4: [3, 5, 7, 10, 1],
5: [7, 2, 4, 9, 1, 3],
6: [1, 8, 3, 0],
7: [3, 5, 10, 0, 2, 4],
8: [9, 2, 6, 3],
9: [8, 1, 3, 5, 10, 0],
10: [3, 7, 9, 4, 1],

I po każdej narysowaniu każdej krawędzi ustawiałem graf tak żeby był planarny.
W pewnym momencie doszedłem do takiej sytuacji:

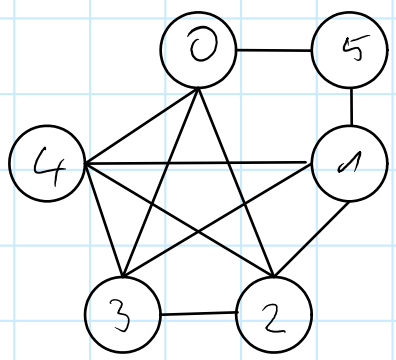


W tym momencie chciałem połączyć wierzchołki 8 i 2. Okazało się że nie ma takiej możliwości żeby ten graf był planarny, ponieważ wierzchołek 2 musi poruszać się w przestrzeni ograniczonej wielobokiem złożonym z wierzchołków 0,1,5,7, a nie ma możliwości, żeby do tego wieloboku włożyć wierzchołek 8, ponieważ jest on połączony do wierzchołka 9. Próbując wrzucić wierzchołki 9,10,3,4 do ww. wielokąta, w każdym przypadku dochodzimy do kolizji któregoś z połączeń z wierzchołkiem drugim, co pokywa wszystkie możliwości.

Dowód z Twierdzenia Kuratowskiego:
Graf jest nieplanarny, ponieważ w grafie występuje podgraf $K_{3,3}$



Założenie do tw. Kuratowskiego:
Na potrzeby twierdzenia, możemy zakładać że dwa wierzchołki są połączone, jeżeli są połączone pośrednio np.



Ten graf jest nieplanarny ze względu na występowanie podgrafu K_5 z pośrednim połaczeniem. Możemy zakładać że wierzchołki 0 i 1 są połączone, ponieważ występuje pośrednie połączenie przez wierzchołek 5.

Cz.4

piątek, 28 maja 2021 12:50

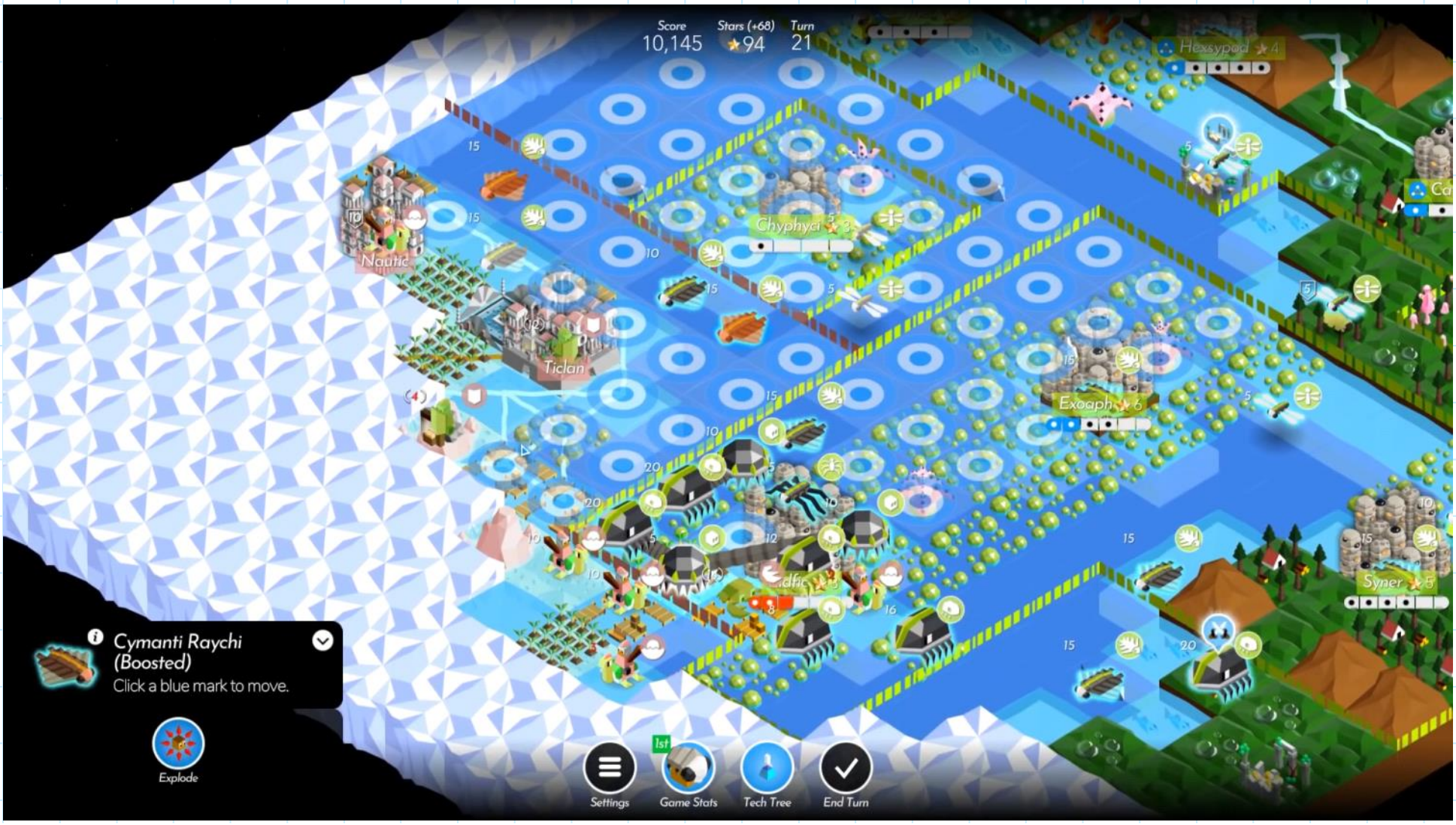
Zaimplementuj algorytm Dijkstry (10pkt)

Przeanalizuj powyższy algorytm: jakie problemy rozwiązuje, konkretne przykłady wykorzystania, z jakich metod korzysta się obecnie do rozwiązywania tych problemów (4pkt)

Algorytm Dijkstry można wykożystać do wyznaczenia kosztu dostania się do danego miejsca oraz do wyszukania najoptymalniejszej ścieżki.
Przykład: Mamy grę planszową, w której gracze sterują pionkami ustawionymi na planszy złożonej z sześciokątów. Ta plansza jest trójwymiarowa. Koszt przejścia z jednego sześciokąta na drugi jest zależny od różnicy wysokości między poprzedzającym a następującym sześciokątem.
Innym przykładem może być wirtualna gra terenowa jak na poniższym obrazku:

Jak widać na uchwyconym momencie, cena przejazdu po terenie jest niższa niż przez wodę. Ten szczegół, oraz poprzedni - z różnicą wysokości terenu - można uchwycić algorytmem dijkstry podczas wyznaczania na które pola można się przemieścić.

Niżej inna gra (Polytopia) w której ten algorytm mógł zostać wykożystany



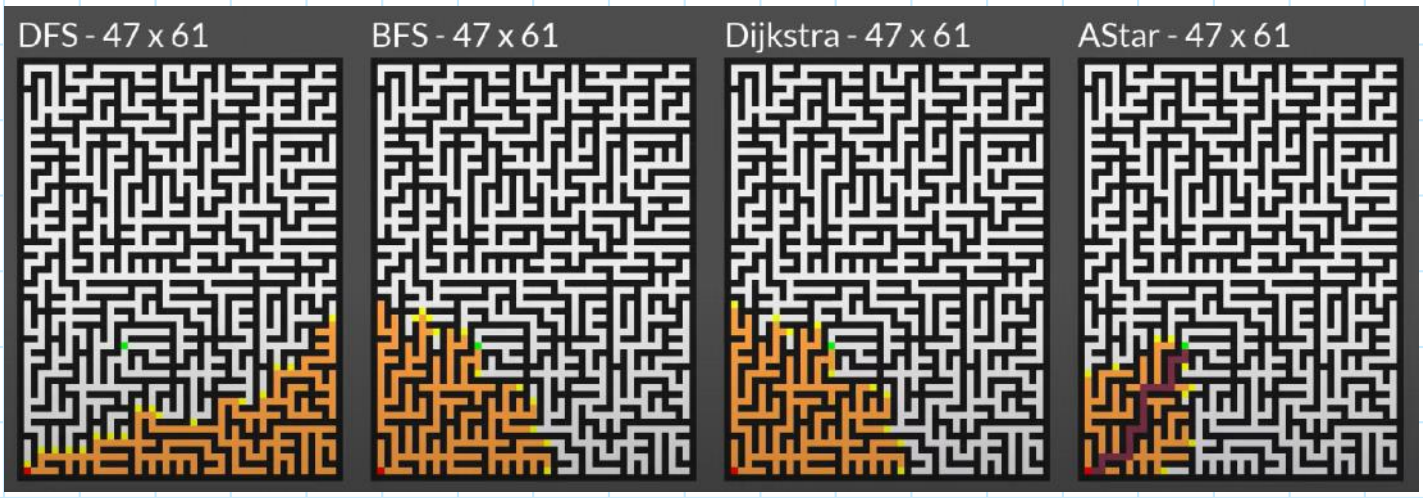
Cz.5

piątek, 28 maja 2021 12:50

Djikstra jest dość aktualnym algorytmem do rozwiązywania w.w. problemów mimo istnienia innych algorytmów zajmujących rozwiązywaniem tych samych problemów (mówimy o problemach optymalnej ścieżki, gdzie ścieżka w każdym punkcie ma dodatni koszt)

Mamy trzech konkurentów na miejsce Dijkstry: DFS, BFS oraz A*
Wszystkie te algorytmy są algorytmami "zakaźnymi", co oznacza że najpierw następuje "sztuczna infekcja" pierwszej komórki, a ona po kolei przydziela wagi sąsiednim komórkom i je "zakarza", tzn poddaje rekurencyjnemu działaniu algorytmu - zakarzone komórki również przydzielają wagi sąsiadom i wywołują algorytm na sąsiadach.
To co je różni natomiast to sposób rozprzestrzeniania się "zakarzenia".

Wyobraźmy sobie że mamy doczynienia z grafem, który dla ułatwienia ma tą samą charakterystykę co plansza z polami kwadratowymi, z tym, że niektóre pola są niedostępne, lub językiem grafów - nie istnieją tam wierzchołki a połączenia są tylko między wierzchołkami, między którymi odległość wynosi 1



Na powyższym obrazku jest przykład takiej planszy oraz uchwycone w trakcie działania algorytmy.

Wracając do sposobu zakarzania. Załózmy że algorytmy w pierwszej kolejności infekują w lewo, następnie w dół, dalej w prawo i w górę.
DFS rozprzestrzenia się po komórkach najwcześniej zainfekowanych, co oznacza że w zależności które kierunki się wybierze w algorytmie, tak algorytm będzie preferował konkretne kierunki nad inne (w naszym przypadku bedzie się poruszał maksymalnie w lewo i w doł)
BFS rozprzestrzenia się po komórkach ostatnio dodanych, co sprawia że w otwartym terenie, rozwija się spiralą dookoła środka.

I teraz teoretycznie najlepszy algorytm z całej czwórki - A*. Algorytm przy rozprzestrzenianiu się preferuje komórki o najmniejszej odległości manhatańskiej do celu. Jeżeli chodzi o ilość infekcji, zdecydowanie wygrywa nad wszystkimi algorytmami, tylko można się zastanawiać czy dodatkowa suma i dwie różnice poddane wartości bezwzględnej nie kompensują zysku z kierunku przeszukiwania, bo czemu nie pójść o krok dalej i uwzględniać prawdziwą odległość (albo i oszukaną, bo porównując tylko wartości między sobą możemy pominąć pierwiastek i operować tylko na $\Delta x^2 + \Delta y^2$).

Moim skromnym zdaniem Dijkstra jest lepsza niż ten algorytm z tej samej przyczyny co powód przez który niektóre super algorytmy do sortowania wcale nie są takie fajne.
Zdarza mi się stawać przed wyborem między quickSortem, mergeSortem, sortowaniem przez wstawianie, sortowaniem kubełkowym, sortowaniem przez konstruowanie drzew wszelkiego rodzaju i w sumie wybór jest prosty. Większość problemów wcale nie jest taka duża, a potrzebuję algorytmu, który będzie działał wystarczająco dobrze, oraz nie będzie wymagał godzin debugowania, co pociąga za sobą wymóg prostoty kodu. I jest algorytm który spełnia aspekty praktyczne znacznie lepiej niż pozostałe. Kwestją kilku minut oraz mniej więcej 7 prostych linijek jest napisanie najbardziej przyjaznego algorytmu sortującego, działającego zwykle za pierwszym razem - bubbleSorta.

Przechodząc do podsumowania, Djikstra jest aktualnie najlepszym algorytmem do wyznaczania optymalnej ścieżki przy dodatnich wartościach ścieżek, ponieważ większość przypadków jest przeciętna, a Djikstra idelanie się do takich nadaje dzięki zachowaniu balansu między czasem wykonywania, złożonością pamięciową oraz poziomem skomplikowania kodu.