Algorytm Dijkstry Znajdowanie najkrótszej drogi w labiryncie. OMP

Paweł Sawicki

1 Wprowadzenie

Algorytm Dijkstry, opracowany przez holenderskiego informatyka Edsgera Dijkstre, służy do znajdowania najkrótszej ścieżki z pojedynczego źródła w grafie o nieujemnych wagach krawedzi.

2 Działanie

Majac dany graf z wyróżnionym wierzchołkiem (źródłem) algorytm znajduje odległości od źródła do wszystkich pozostałych wierzchołków. Łatwo zmodyfikować go tak, aby szukał wyłacznie (najkrótszej) ścieżki do jednego ustalonego wierzchołka, po prostu przerywajac działanie w momencie dojścia do wierzchołka docelowego, badź transponujac tablice incydencji grafu. Algorytm Dijkstry znajduje w grafie wszystkie najkrótsze ścieżki pomiedzy wybranym wierzchołkiem a wszystkimi pozostałymi, przy okazji wyliczajac również koszt przejścia każdej z tych ścieżek. Algorytm ten jest przykładem algorytmu zachłannego.

3 Algorytm

Nazwijmy wierzchołek startowy v0. Niech odległość wierzchołka Y bedzie odległościa od wierzchołka v0 do wierzchołka Y. Algorytm przydzieli im odległości poczatkowe, a potem te odległości poprawi.

1. Przydziel każdemu wierzchołkowi odległość(d):d[v0]=0, dla reszty d[vi]=inf.

2. Zaznacz wszystkie wierzchołki jako nieodwiedzone. Ustaw v0 jako aktualny wierzchołek. Stwórz tablice nieodwiedzonych wierzchołków.

3.Dla aktualnego wierzchołka rozważ nieodwiedzonych sasiadów i porównaj ich wagi. Wybierz najmniejsza. Nastepnie ustaw wierzchołek z najmniejsza waga jako aktualny i usuń z wierzchołków nieodwiedzonych.

4. Kiedy zostana rozważeni wszyscy sasiedzi wierzchołka, ustaw go jako odwiedzony i usuń z nieodwiedzonych. Odwiedzony wierzchołek nie bedzie wiecej sprawdzany.

5. Jeżeli wierzchołek docelowy jest ustawiony jako odwiedzony(planujac droge pomiedzy dwoma konkretnymi wierzchołkami) albo jeżeli jego waga wynosi nieskończoność to koniec. Algorytm został zakończony.

6. Wybierz nieodwiedzony wierzchołek, który ma najmniejsza wage i ustaw jako aktualny wierzchołek, a potem wróć do kroku trzeciego.

4 Rozwiazanie

Program sekwencyjny został napisany tak aby wykonywał normalny algorytm dijkstry(odległość od wierzchołka poczatkowego do każdego wierzchołka w grafie). Jedynie drukowana jest odległość od wybranego wierzchołka v1 do wybranego wierzchołka końcowego. Program równoległy jest taki sam z wyjatkiem drukowania, tutaj drukowane sa dodatkowo wszystkie odległości. Dodatkowo jest wydruk od v1 do vk. Jedynie kroki algorytmu dijkstry sa równoległe, wczytywanie z pliku i drukowanie sa sekwencyjne.

5 Dane wejściowe

Generator po otrzymaniu ilości wierzchołków jakie chcemy mieć w grafie(labiryncie) generuje nam krawedzie pomiedzy wierzchołkami. ./a.out liczbav vkoncowe iloscscian plikwyjsciowy

```
psawicki@sigma:~/ITHPC$ ./a.out 50 47 3 dane.txt
psawicki@sigma:~/ITHPC$ cat dane.txt
1
47
200
1 2 1
1 0 1
1 11 1
2 3 1
2 1 1
2 12 1
```

Fig. 1: Przykład generowania

Labirynty generowane sa w formie:

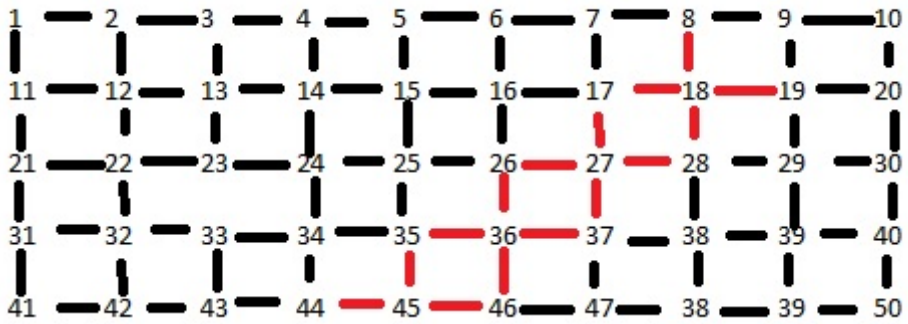

Fig. 2: Przykładowy labirynt

Plik dane.txt (plik w folderze obok).W miejcach gdzie sa czerwone połaczenia wagi wynosza 100, w czarnych 1. Wagi 100 ‘symuluja’ ściany labiryntu. Na takim labiryncie wykonano pierwsze 2 testy.

6 Pomiar Czasu

Każdy test odpalany był 5 razy i do tabelki został wpisany średni wynik.

Test	4	watki Sigma (8 watków)	32 watki	Xeon Phi 240 watków
	Time:	Time:	Time:	Time:
Sekwencyjnie 169 dróg	Time:	Time:	Time:	Time:
	0.000408515	0.001370413	0.000329656	0.001110299
	Read:	Read:	Read:	Read:
	0.000152748	0.000112501	0.000144364	0.0005944
	Steps:	Steps:	Steps:	Steps:
	0.000035639	0.000016141	0.000030662	0.0000468
OMP 50	Time:	Time:	Time:	Time:
	0.019825061	0.033928341	0.017841893	0.278055531
	Read:	Read:	Read:	Read:
	0.000111322	0.000099244	0.000144509	0.000701645
	Steps:	Steps:	Steps:	Steps:
	0.019491082	0.032262697	0.017456805	0.250724569
OMP 100	Time:	Time:	Time:	Time:
	0.001321087	0.00175629700	0.043707225	0.273716571
	Read:	Read:	Read:	Read:
	0.000245507	0.00012285100	0.000116262	0.000977124
	Steps:	Steps:	Steps:	Steps:
	0.000408094	0.000389783	0.02523804	0.271734131
OMP 1000	Time:	Time:	Time:	Time:
	0.018177334	0.134693921	0.028073719	0.292792703
	Read:	Read:	Read:	Read:
	0.002951574	0.002354198	0.000876282	0.008462411
	Steps:	Steps:	Steps:	Steps:
	0.014092035	0.110725065	0.009335235	0.253203801
OMP 10000	Time:	Time:	Time:	Time:
	0.41508565	0.193165775	0.036099128	0.424844348
	Read:	Read:	Read:	Read:
	0.013204211	0.008954283	0.000858164	0.086286142
	Steps:	Steps:	Steps:	Steps:
	0.393656946	0.160013222	0.021397569	0.258833177
OMP 10000	Time:	Time:	Time:	Time:
	39.7326325639999993	12.0788551389999999	0.043406191	1.8360153440000002
	Read:	Read:	Read:	Read:
	0.097112304	0.09575919	0.008435412	0.9781656870000001
	Steps:	Steps:	Steps:	Steps:
	39.57622549700	11.9192767530000001	0.013507856	0.417627523
OMP 500000	Time:	Time:	Time:	Time:
	83.63712682	15.02447536	0.593838284	1.76472193
	Read:	Read:	Read:	Read:
	0.189289604	0.09248116	0.088157858	0.901385709
	Steps:	Steps:	Steps:	Steps:
	83.3847826070000053	12.2416325	0.461420997	0.427203886
OMP 1000000	Time:	Time:	Time:	Time:
	141.26182324	21.02447536	0.971426213	1.8761751
	Read:	Read:	Read:	Read:
	0.241487416	0.15153216	0.1371435	0.9813547
	Steps:	Steps:	Steps:	Steps:
	139.9571	19.9012991	0.8161478	0.443157231