Sprawozdanie - Laboratorium 09 PAMSI

Artur Gasiński — 218685

09.05.2016

1 Zadanie

1. Implementacja wyszukiwania najkrótszej ścieżki w grafie ważonym metodą Branch & Bound.
2. Pomiar czasów wyszukiwania najkrótszej ścieżki algorytmami standardowym i ulepszonym ("with extended list") dla n wierzchołków w grafie: $n = 10, 10^2, 10^3, 10^6, 10^7$;

2 Wykonanie

1. Struktura programu:
   - interfejs IRunnable,
   - interfejs IGraph (zawiera podstawowe metody ADT grafu),
   - klasa Graph, implementująca interfejs IGraph:
     - graf zbudowany jest z listy wierzchołków reprezentowanych przez liczby naturalne,
     - krawędzie są reprezentowane listą obiektów typu Edge, przechowującymi wierzchołki startowy i końcowy oraz wagę krawędzi (przyjęto dopuszczalne wagi od 1 do 10)
     - połączenia między wierzchołkami reprezentowane są w tablicy list sąsiedztwa dla każdego wierzchołka,
     - w liście zapisane są obiekty typu Voisin, przechowujące sąsiedni wierzchołek i wagę krawędzi,
     - założono, że graf ma być spójny (tzn. nie ma ani jednego węzła nie połączonego z resztą grafu), nieskierowany (krawędzie między dwoma wierzchołkami prowadzą w obie strony) oraz ważony, nie ma też krawędzi wychodzących z wierzchołka i wracających do niego samego.
   - klasa Graph-Test, implementująca intefejs IRunnable dla klasy Graph,
   - pomocnicze interfejsy i implementacje listy dwukierunkowej, kolejki i stosu (odpowiednio klasy IList, BList, IQueue, Kolejka, IStack, Stos),
   - klasy Stopwatch i AdvancedStopwatch, wykonujące pomiar czasów korzystając z funkcji gettimeofday(),
   - funkcja główna, zarządzająca kolejnością wykonywania zadań.

2. Idea algorytmu Branch&Bound:

   - utwórz listę P przechowującą obiekty ścieżek typu Path (zawiera listę wierzchołków i długość ścieżki),
   - dodaj wierzchołek startowy do listy P,
   - dopóki lista P nie jest pusta i ostatni wierzchołek nie jest założonym końcowym, wykonuj:
     - weź ścieżkę z listy P,
     - utwórz x nowych ścieżek poprzez wydłużenie jej do wszystkich x sąsiadów ostatniego wierzchołka w ścieżce,
     - odrzuć ścieżki, które zawierają pętle,
     - dodaj nowe ścieżki do pomocniczej tablicy dynamicznej,
     - wyszukaj ścieżkę o najmniejszej długości (najmniejszej sumie wag wszystkich krawędzi),
     - przenieś najkrótszą ścieżkę z tablicy do listy.
   - Modyfikacja "with extended list" polega na przechowywaniu informacji (w tym przypadku w tablicy) o sprawdzeniu danego wierzchołka, tzn. że przeszukano już ścieżki zawierające go i jego sąsiadów. Jeżeli wierzchołek został oznaczony jako wydłużony ("extended"), jest pomijany przy najstępnych przejściach pętli algorytmu. To redukuje liczbę sprawdzanych ścieżek i tym samym oszczędza czas.

3. Pomiary wykowano w następujący sposób:
   Dla każdego n wywoływano metodę $Prepare(n)$, która dodawała n wierzchołków do grafu i tworzyła między nimi spójne powiązanie. Dokonywano tego w następujący sposób: tworzono tymczasową tablicę z numerami wierzchołków i następnie losowo zamieniano komórki tablicy. Następnie dodawano krawędzie zgodnie z wylosowaną kolejnością (wagi krawędzi generowano losowo). Na koniec przygotowania, metoda Prepare() generowała $2n$ losowych krawędzi o losowych wagach w zakresie od 1 do 10. Założono, że dwa wierzchołki może łączyć tylko jedna krawędź, aby nie było sytuacji, kiedy istnieją dwie ścieżki o różnych wagach.

   W tak przygotowanym grafie wyszukiwano najkrótszą ścieżkę między wierzchołkami 0 i wybranym losowo - najpierw standardowym algorytmem Branch & Bound, a później wersją "with extended list". Każdy algorytm wyświetlał znalezioną najkrótszą ścieżkę wraz z jej wagą, oraz liczbę wszystkich sprawdzonych ścieżek. Dla obu przejść zmierzono czas działania. Pomiary przeprowadzono w seriach po 6 razy. Za każdym razem losowano nowy wierzchołek końcowy szukanej najkrótszej ścieżki, aby pomiary czasu były bardziej wiarygodne. Zmodyfikowano nieco rozmiary problemów: $n = 10, 10^2, 10^3, 10^4, 10^5$.

Tabela 1: Branch&Bound

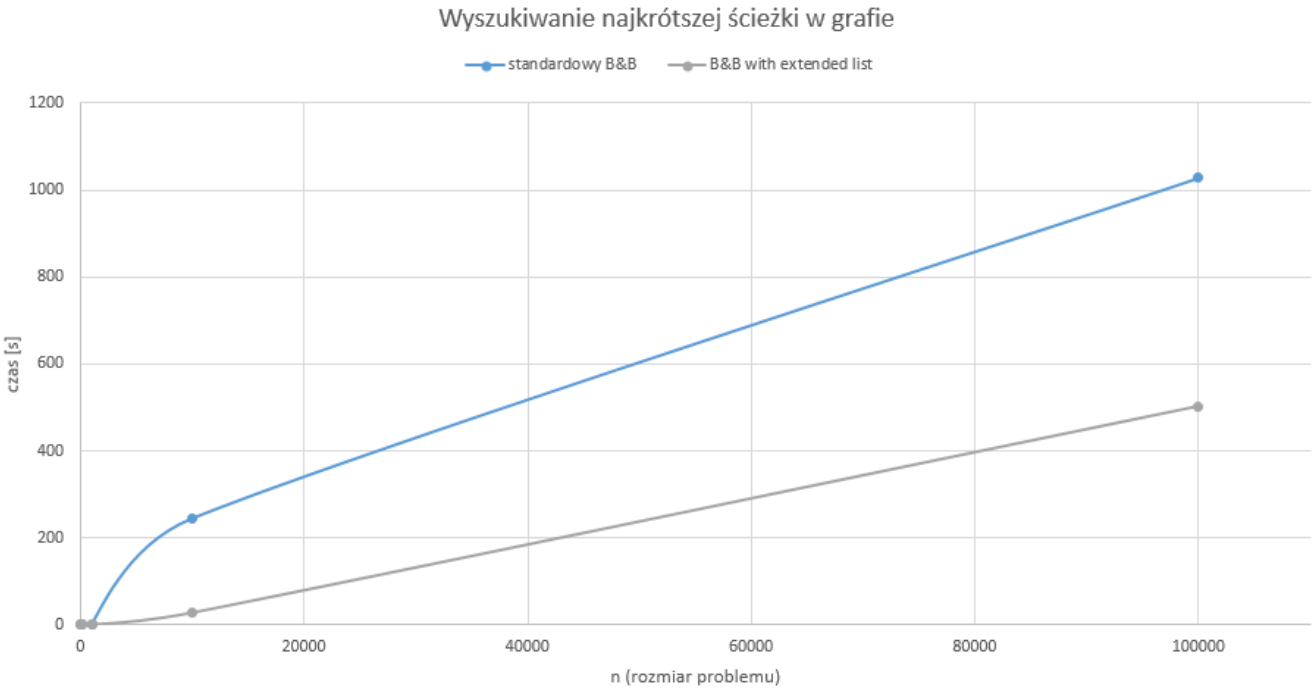
n	10	10^2	10^3	10^4	10^5
czas [s]	0,00004383	0,00352616	0,24834869	244.27972168	1028.96600553

Tabela 2: Branch&Bound with extended list

n	10	10^2	10^3	10^4	10^5
czas [s]	0.00003782	0.00163018	0,09955452	27,136112680	502.48272431

Tabela 3: Liczba sprawdzanych ścieżek

N	B&B	B&BEL	różnica [%]
10	4	4	0
	23	23	0
	4	4	0
	7	7	0
	23	23	0
	19	19	0
100	490	390	20.41
	28	28	0
	689	362	47.46
	689	362	47.46
	183	160	12.56
	48	48	0
1000	7508	3554	52.66
	8347	3710	55.55
	203	203	0
	673	650	3.41
	889	849	4.49
	1086	976	10.12
10000	93660	37561	59.89
	40541	25728	36.53
	31859	22559	29.19
	329291	47943	85.44
	102398	39019	61.89
	79699	36071	54.74
100000	180737	142591	21.10
	158595	128076	19.24
	589172	305048	48.22
	242813	177509	26.89
	352755	226138	35.89
	13857	13857	0



3 Wykres

4 Wnioski

Wyniki pomiarów pokazują, jak nieefektywne są algorytmy szukające najkrótszej ścieżki w grafie. Standardowy algorytm Branch&Bound wykonuje bardzo dużo obliczeń, czasem szukając niepotrzebnie drogi przez sprawdzony już wierzchołek. Odbija się to na czasie działania - jest znacznie dłuższy niż przy metodzie ulepszonej. Różnice w liczbach sprawdzonych ścieżek sięgają czasem nawet 2 rzędów wielkości, a to musi się odbić negatywnie na czasie działania.