Sabina Miśkiewicz200577
Wt 18.00

Sprawozdanie z laboratorium projektowanie algorytmów i metod sztucznej inteligencji

Algorytmy do obliczania minimalnego drzewa rozpinającego

Doświadczenia zostały przeprowadzone na komputerze o następujących parametrach:

- System: 64-bitowy Windows 7
- Procesor: Pentium® Dual-Core CPU T4500 @ 2.30GHz 2.30GHz
- Pamięć RAM: 3,00 GB

Omówienie algorytmów:

Kruskala

Algorytm jest oparty o metodę zachłanną. Polega na łączeniu wielu poddrzew w jedno za pomocą krawędzi o najmniejszej wadze. W rezultacie powstałe drzewo będzie minimalne. Na początek należy posortować wszystkie krawędzie w porządku niemalejącym. Po tej czynności przystępuje się do tworzenia drzewa. Proces ten nazywa się rozrastaniem lasu drzew. Wybieramy krawędzie o najmniejszej wadze i jeśli wybrana krawędź należy do dwóch różnych drzew należy je scalić (dodać do lasu). Krawędzie wybieramy tak długo, aż wszystkie wierzchołki nie będą należały do jednego drzewa.

Prima

Algorytm oparty o metodę zachłanną. Aby zbudować minimalne drzewa rozpinające musimy wybrać wierzchołek, od którego rozpoczniemy. Dodajemy wierzchołek do drzewa, a wszystkie krawędzie łączące umieszczamy na posortowanej według wag liście. Następnie zdejmujemy z listy pierwszą krawędź, czyli tą, do której przypisana jest najmniejsza waga. Sprawdzamy, czy drugi wierzchołek tej krawędzi należy do tworzonego drzewa. Jeżeli tak, to nie dodajemy takiej krawędzi, ponieważ oba jej wierzchołki znajdują się już w drzewie, porzucamy krawędź i pobieramy z listy następną. Jeżeli jednak wierzchołka nie ma w drzewie, to należy dodać krawędź do drzewa, by wierzchołek ten znalazł się w drzewie rozpinającym. Następnie dodajemy do posortowanej listy wszystkie krawędzie łączące z dodanym wierzchołkiem i pobieramy z niej kolejny element.
Zawsze dodajemy do drzewa krawędź o najmniejszej wadze, osiągalną (w przeciwieństwie do algorymu Kruskala) z jakiegoś wierzchołka tego drzewa.
Kluczową czynnością, z punktu widzenia wydajności algorytmu jest zaimplementowanie listy posortowanej, gdyż po każdym dodaniu krawędzi, lista musi być nadal posortowana.

Omówienie przebiegu eksperymentu z przedstawieniem wykresów i tabel

Kruskala

Złożoność czasowa wynosi O(E log(V)).

	czas
v	g=50%
256	27084,58
512	68359,38
1024	Komp poległ
2048	Komp poległ
4096	Komp poległ

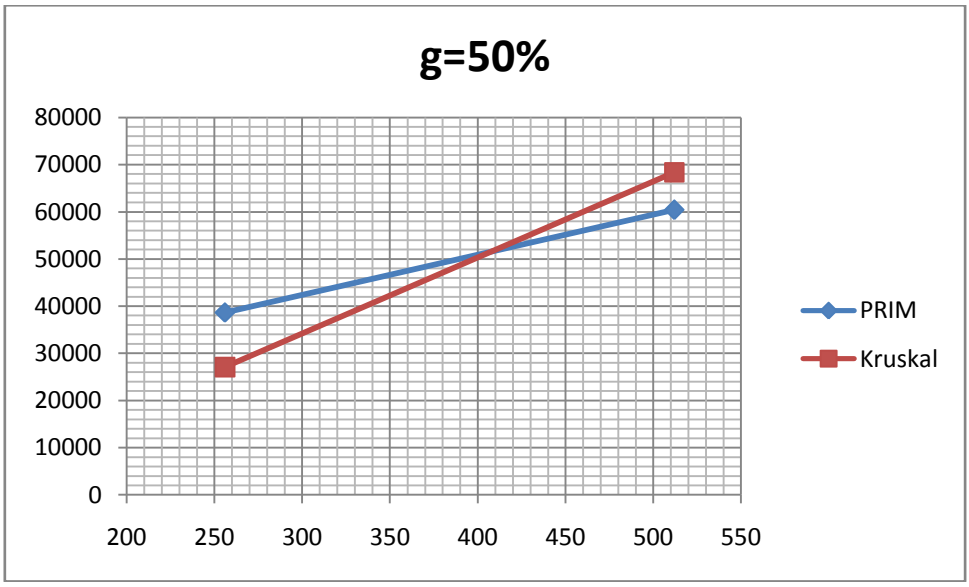
v= 256	krusz
g	czas
20	23065,74
40	25254,25
60	30054,11
80	39523,34
100	Program poległ

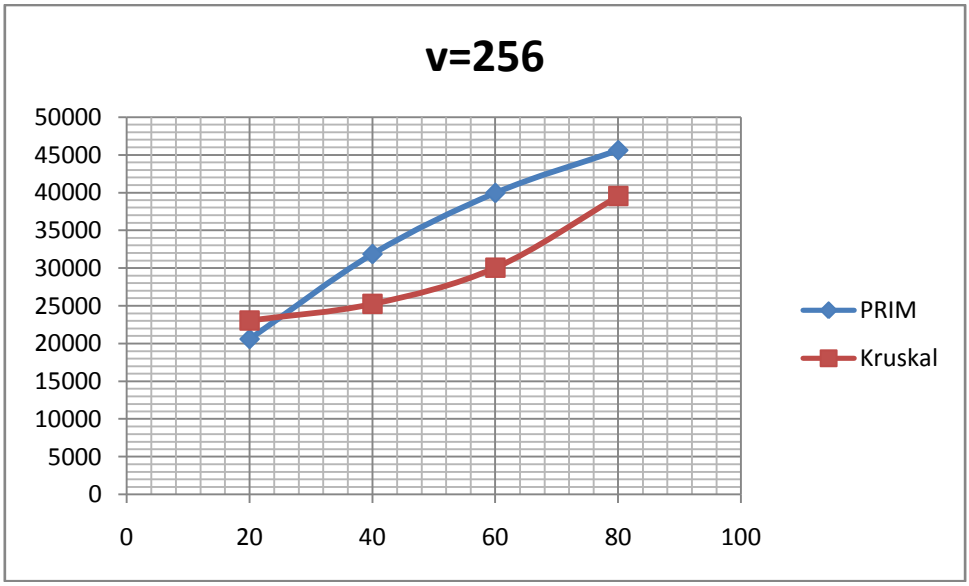
Prima

Złożoność czasową szacuje się na O(E log(V)).

	czas
v	g=50%
256	38654,48
512	60411,77
1024	Komp poległ
2048	Komp poległ
4096	Komp poległ

v= 256	
g	
20	20582,43
40	31854,28
60	39959,95
80	45622,02
100	Komp poległ





Wady i zalety

Kruskala

Zalety: Nie ma dużych wymagać pamięciowych. W każdym kroku algorytmu podejmowana jest optymalna na danym etapie decyzja dotycząca wyboru elementu, który będzie częścią rozwiązania wynikowego. Ten element nie jest już nigdy później zmieniany. Nie musi pamiętać struktury całego grafu.
Wady: W algorytmach zachłannych często płacimy daleką od optymalności postacią rozwiązania problemu.

Prima

Zalety: W każdym kroku algorytmu podejmowana jest optymalna na danym etapie decyzja dotycząca wyboru elementu, który będzie częścią rozwiązania wynikowego. Ten element nie jest już nigdy później zmieniany. Zasadniczo nie ma dużych wymagań pamięciowych.
Wady: W algorytmach zachłannych często płacimy daleką od optymalności postacią rozwiązania problemu.

Wioski i posumowanie

Każdy algorytm ma swoje wady i zalety. Ja zrobiłam algorytmy tylko na listach. W obu algorytmach do sortowania danych używam sortowania przez scalanie o złożoności obliczeniowej O (n log n). Dla dużej ilości danych czasy wzrastały do bardzo długich.

Wykorzystane materiały i źródła:

- Thomas H. Cormes, Charles E. Leiserson, Ronald L. Rivest “Wprowadzenie do algorytmów” wydanie czwarte
- Adam Drozdek “C++ algorytmy I struktury danych” wydawnistwa Helion