Sprawozdanie z ćwiczenia laboratoryjnego

Tomasz Mikołajewski

2014

Spis treści

1 Wprowadzenie **1**

2 Wyniki pomiarów **2**

2.1 Breadth-First-Search 2

2.2 Depth-First-Search 3

2.3 Best-First-Search 3

2.3.1 Wyszukiwanie zawsze wybierające najmniejszą wartość kroku. . . 3

2.3.2 Wyszukiwanie bez rozważania wagi ostatniego kroku. 4

2.4 Złożoność obliczeniowa algorytmu 4

2.5 A* 7

3 Wnioski **9**

1 Wprowadzenie

Niniejszy dokument powstał w ramach przedmiotu Projektowanie Algorytmów i Metod Sztucznej Inteligencji. Jest on rezultatem przeprowadzonych ćwiczeń w laboratorium oraz pracy wykonanej w domu.

W dokumencie tym podsumowano algorytmy działające na grafach. Sprawdzane było działanie algorytmów:

- Breadth-First-Search (BFS)
- Depth-First-Search (DFS)
- Best-First-Search (BestFS)
- A*

Wykonano analizę czasu potrzebnego do wyszukania konkretnego elementu w grafie przez poszczególne algorytmy.

2 Wyniki pomiarów

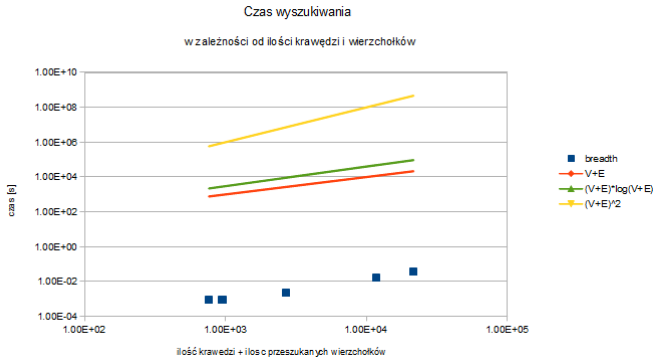
2.1 Breadth-First-Search

Jest to algorytm wykonujący przeszukiwanie zaczynając od sąsiadów najbliższych elementowi początkowemu. Wykorzystuje on listę typu FIFO, na której gromadzone są wierzchołki, do których należy się odwołać. Podczas wykonywania algorytmu zapisywana jest odległość pomiędzy punktem startowym, a danym elementem. Jest to przydatna informacja np: przy szukania najkrótszej drogi pomiędzy punktami.

Dane zebrane podczas pomiarów zostały przedstawione w Tablica 1. Na ich podstawie wykreślono Rysunek 1.

Tablica 1: Przeszukiwanie wszerz

ilość elementów	ilosc krawedzi	czas średni [s]
310	640	0,00089
250	512	0,00087
3900	7800	0,01588
900	1800	0,0022
7200	14400	0,03517



Rysunek 1: Przeszukiwanie wszerz

Jak widać na załączonym wykresie złożoność algorytmu przeszukiwania BFS jest liniowa. Warto również zauważyć, że zależy ona zarówno od ilości krawędzi jak i wierzchołków.

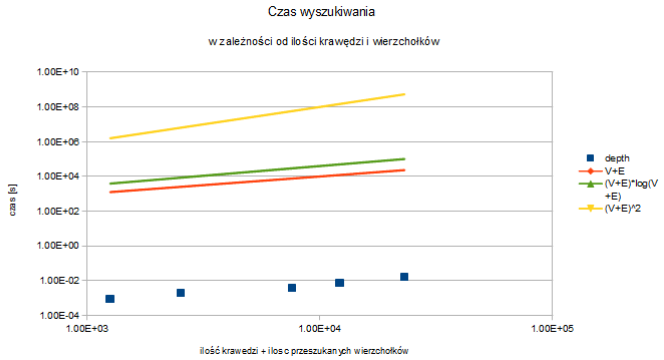
2.2 Depth-First-Search

Algorytm wykonujący przeszukiwanie w głąb polega na wywoływaniu sąsiadów wierzchołków aż do momentu osiągnięcia krańca grafu lub powrotu do wcześniej odwiedzonego wierzchołka. Teoretycznie powinien on mieć złożoność podobną do wcześniej omawianego algorytmu.

Zebrane dane zostały przedstawione w Tablica 2 , a następnie zwizualizowane na Rysunek 2

Tablica 2: Przeszukiwanie wszerz

ilość elementów	ilosc krawedzi	czas średni [s]
17850	157700	0.7191
11800	51600	0.0958
18270	420500	4.19
5000	18100	0.0165
2800	10200	0.0093



Rysunek 2: Złożoność algorytmu

Podobnie jak poprzedni algorytm, również DFS posiada złożoność liniową zależną zarówno od ilości wierzchołków jak i krawędzi.

2.3 Best-First-Search

2.3.1 Wyszukiwanie zawsze wybierające najmniejszą wartość kroku.

Jest to rodzaj wyszukiwania najprostszy pod względem założeń. Algorytm ten zawsze wybiera ścieżkę o najniższym koszcie, a jeśli obrana droga nie doprowadzi go do celu to wybierana jest kolejna ścieżka wychodząca z ostatniego węzła (lub w przypadku braku innych krawędzi do sprawdzenia z poprzedniego węzła). Algorytm ten wykorzystuje listę

na której znajdują się kolejne możliwe kierunki ruchu w grafie. W programie, na podstawie którego zostało wykonywane sprawozdanie, użyto listy LIFO. Aby było to możliwe dodawano kolejne ścieżki od najmniej korzystnej do najbardziej korzystnej po każdorazowej zmianie wierzchołka. Rezultat algorytmu jest opisany na podstawie poniższego grafu (Rysunek 3).

Należało znaleźć drogę pomiędzy g i k.

Znaleziona droga to: g->f->o->u->a->p->k Koszt drogi to 342.

2.3.2 Wyszukiwanie bez rozważania wagi ostatniego kroku.

Algorytm ten kończy swoje działanie, gdy z wierzchołka w którym się znajduje osiągalny jest wierzchołek docelowy. Koszt ostatniego kroku jest dodawany do ogólnego kosztu całej drogi, lecz w tym miejscu następuje wyłamanie się z ogólnej zasady algorytmu polegającej na wybieraniu zawsze drogi o najniższym koszcie. Niesie to ze sobą konsekwencje omówione w dalszej części sprawozdania.

Sytuacje przy których algorytm okazał się skuteczniejszy.

Jeżeli pomija się istotność ostatniego kroku to nie wykonuje się dodatkowych operacji polegających na dalszym przeszukiwaniu grafu. Oznacza to, że wyszukana ścieżka może być krótsza od znalezionej w algorytmie trzymającym się zasady wybierania najtańszej ścieżki do samego końca. Statystycznie krótsza ścieżka oznacza mniejszy koszt poniesiony na drodze pomiędzy punktami. Omawiana sytuacja przedstawiona jest na poniższym grafie (Rysunek 4).

W przedstawianym grafie należało znaleźć drogę z p do w.

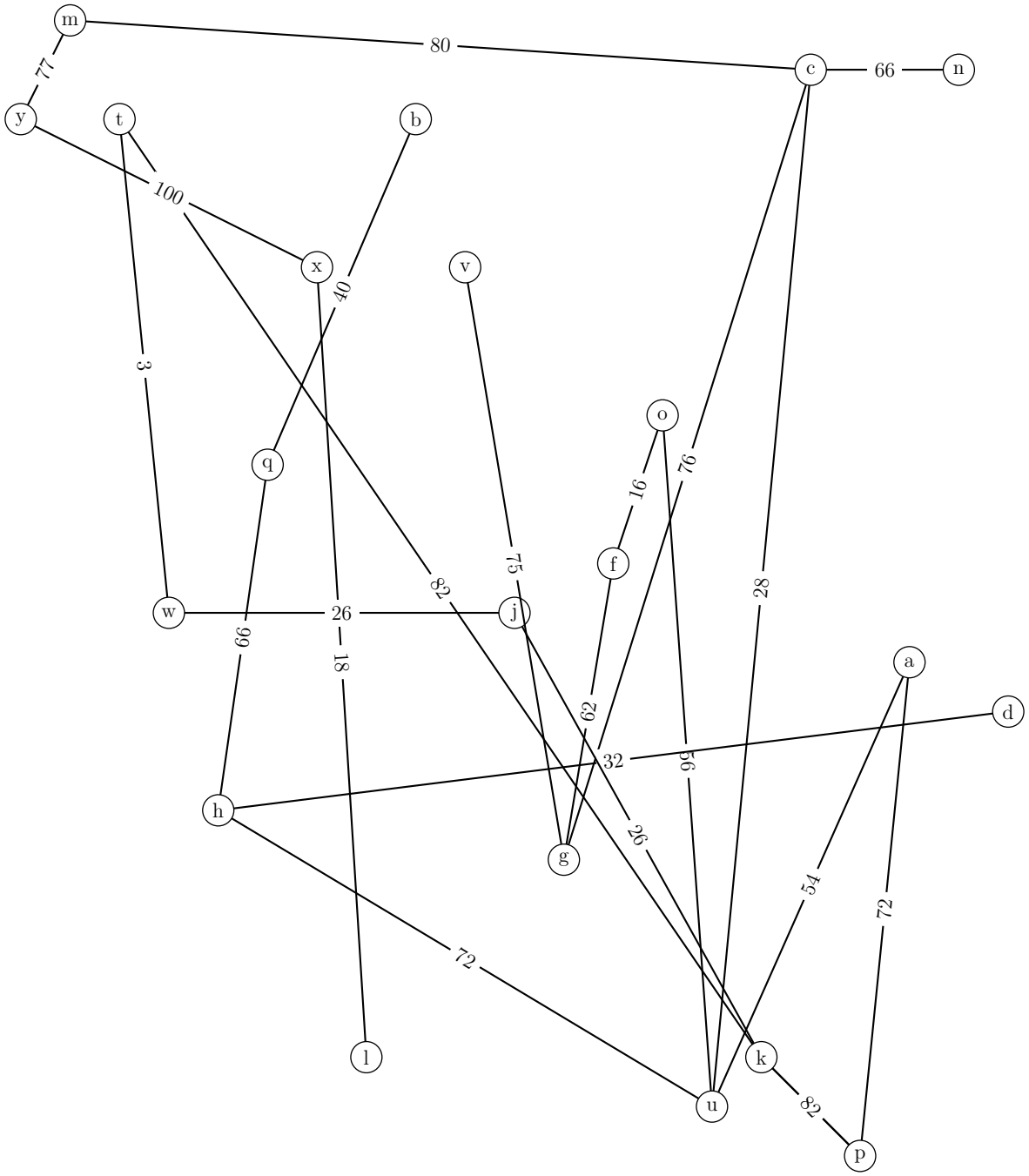
Dla podstawowej wersji wyszukiwania (z uwzględnianiem kosztu ostatniego kroku) znaleziono drogę: p->u->m->l->s->a->w Łączny koszt tej drogi to: 156

Jeżeli zastosuje się algorytm pomijający wagę ostatniego kroku to otrzyma się ścieżkę: p->u->m->l->w Łączny koszt tej drogi to: 147

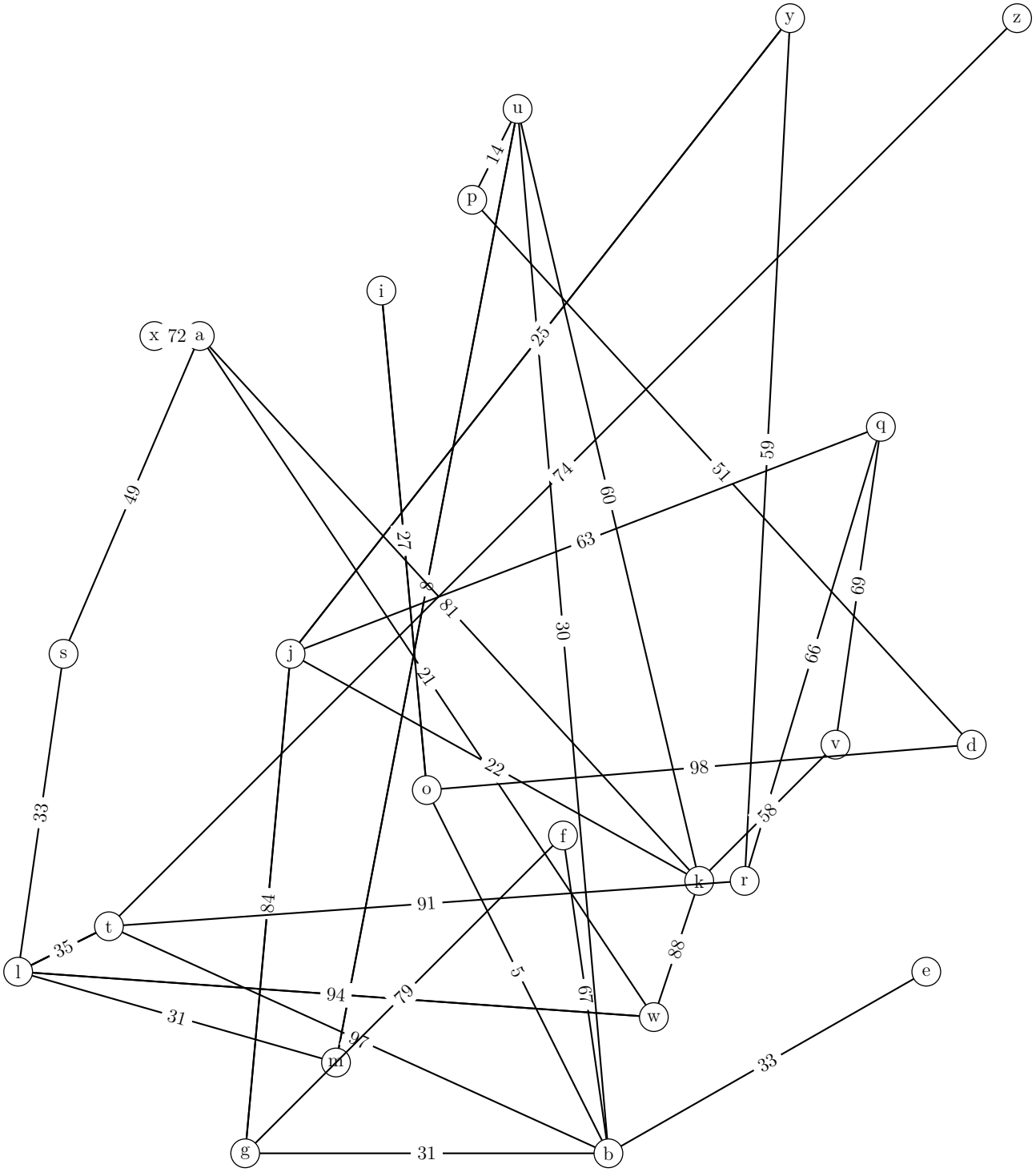
2.4 Złożoność obliczeniowa algorytmu

Ze względu na zależność opisaną w poprzednim punkcie postanowiono wykonywać symulacje dla określonej gęstości grafu. Przyjęto gęstość wynoszącą około 50 procent. Następnie wykonano pomiary zamieszczone w tabeli na Tabela 3

Rysunek 3: Najprostszy algorytm

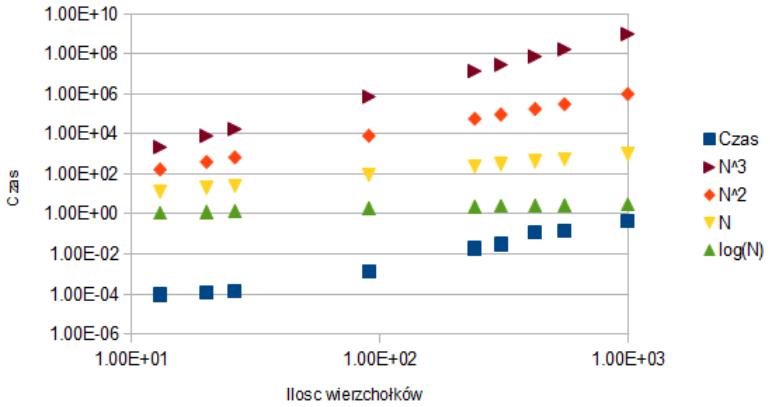


Rysunek 4: Skuteczniejszy



Tablica 3: BestFS

Ilość wierzchołków	Czas [s]
26	0.000146
20	0.000121
13	0.000094
90	0.001312
306	0.030253
552	0.145129
240	0.019125
420	0.11745
992	0.439



Rysunek 5: Złożoność algorytmu

Złożoność obliczeniowa programu ulega zmianie. Dla niewielkiej ilości wierzchołków jest ona liniowa, a wraz ze wzrostem rozmiaru problemu przechodzi w kwadratową. Jednakże zmiana ta jest spowodowana przejściem złożoności odczytu z tablicy z hashowaniem ze stałej na liniową. Złożoność BestFS jest liniowa.

2.5 A*

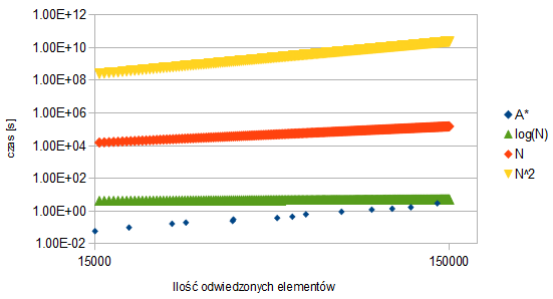
Algorytm ten wykorzystuje przewidywanie pozostałej długości ścieżki w celu optymalizacji wybieranej drogi (tzw. heurystykę). Dzięki zastosowanym obliczeniom kieruje się on od razu w stronę celu i wyszukuje najkrótszą drogę.

Dane zebrane podczas pomiarów zostały przedstawione w Tablica 4.

Na podstawie danych wykreślono wykres zależności czasu wykonywania algorytmu od ilości odwiedzonych pól Rysunek 6

Tablica 4: A*

Kroków	Odwiedzono	Czas [s]
252	15041	0.06
272	18754	0.1
313	24807	0.17
349	27141	0.2
417	36751	0.25
393	36937	0.31
502	49105	0.38
554	54181	0.46
585	74587	0.92
628	90762	1.24
642	103649	1.41
686	116968	1.72
779	138919	3.01
981	215167	4.61



Rysunek 6: Złożoność algorytmu

Jak widać algorytm ten ma złożoność liniową względem odwiedzonych pól.

3 Wnioski

Na podstawie danych, grafów i wykresów można wyciągnąć następujące wnioski:

- wszystkie testowane algorytmy mają podobną złożoność obliczeniową.
- statystycznie BFS spisuje się lepiej od DFS jeżeli najkrótsza droga pomiędzy punktem startowym, a końcowym zawiera małą ilość kroków. Przy stosunkowo dużych wartościach (punktu położone po dwóch ścronach"grafu) DFS statystycznie jest algorytmem wydajniejszym
- jeżeli algorytm A* nie przeszacowuje podczas wykonywania obliczeń związanych z heurystyką to zawsze znajduje najkrótszą ścieżkę (w odróżnieniu od pozostałych)
- po bliższej analizie wykresów widać, że ostatni z omawianych algorytmów (A*) ma lepszą złożoność obliczeniową od pozostałych, ponieważ posiada najmniejszy współczynnik nachylenia prostej
- A* dzięki swoim zaletą znalazł zastosowanie między innymi w grach komputerowych czy w programach GPS.