

# Sprawozdanie I

Przeszukiwanie grafów i planowanie tras przejazdu

**Autor: Konrad Kalita**

## Sekcja "4 Wyszukiwanie najkrótszej ścieżki":

### 1. Statystyki dla problemu bez ograniczeń

<i>Ustawienia</i>	<b>Długość ścieżki</b>	<b>Czas pracy (ms)</b>	<b>Liczba operacji</b>
<i>Breadth-First-Search (domyślny)</i>	16.14	1.90	1243
<i>Dijkstra (domyślny)</i>	16.14	3.55	1614
<i>Best-First-Search (domyślny)</i>	16.14	104.52	1614
<i>A* (domyślny)</i>	16.14	105	1614
<i>Best-First-Search (dx)</i>	16.14	10.91	131
<i>A* (dx)</i>	16.14	32.99	509
<i>Best-First-Search (Manhattan)</i>	16.14	7.07	75
<i>A* (Manhattan)</i>	16.14	7.50	75
<i>Best-First-Search (Diagonal)</i>	16.14	6.48	75
<i>A* (Diagonal)</i>	16.14	9.01	113
<i>Best-First-Search (Euclidean)</i>	16.14	6.78	75
<i>A* (Euclidean)</i>	16.14	9.48	127

### 2. Jak wyglądają definicje heurystyki (w narzędziu on-line):

- Manhattan:  $dx + dy$
- Diagonal:  $(dx + dy) - (0.6 * \text{Math.min}(dx, dy))$
- Euclidean:  $\text{Math.sqrt}(dx * dx + dy * dy)$

### 3. Jaka heurystyka była w stanie doprowadzić algorytmy A\* i Best-First-Search do przejścia nad dłuższą krawędzią w **zadaniu 1**, podpunkt 3?

Przykładowo heurystyka -  $8 * dy$  jest w stanie doprowadzić algorytmy A\* i Best-First-Search do przejścia nad dłuższą krawędzią.

### 4. Statystyki dla problemu z **zadania 2**:

<i>Ustawienia</i>	<b>Długość ścieżki</b>	<b>Czas pracy</b>	<b>Liczba operacji</b>
<i>A* (Euclidean)</i>	36.49	121.27	2252
<i>A* (Manhattan)</i>	36.49	75.61	1300
<i>A* (Diagonal)</i>	36.49	91.72	1466
<i>A* (dx)</i>	36.49	125.18	2252
<i>A* (5*dx)</i>	36.49	84.96	1415
<i>A* (dy)</i>	36.49	140.53	2507
<i>Breadth-First-Search (bidirectional)</i>	36.49	1.18	282

<i>Dijkstra (bidirectional)</i>	36.49	0.93	279
<i>A* (Euclidean, bidirectional)</i>	38.14	10.9	148
<i>A* (Manhattan, bidirectional)</i>	38.14	13.24	191
<i>A* (Diagonal, bidirectional)</i>	38.14	11.84	163
<i>A* (dx, bidirectional)</i>	36.49	10.72	148
<i>A* (5*dx, bidirectional)</i>	38.49	11.17	155
<i>A* (dy, bidirectional)</i>	38.14	21.4	324

5. Czy któraś heurystyka działała w lepiej w trybie jednokierunkowym?

Z moich obserwacji wynika, że nie. Dwukierunkowe algorytmy są znacznie szybsze i wykonują mniej operacji. Są po prostu dużo efektywniejsze. Jedyną wadą jest, że czasem znaleziona droga jest dłuższa niż przy użyciu algorytmu jednokierunkowego. Tak zwany trade-off pomiędzy czasem wykonania a jakością rozwiązania.

6. Zaproponuj najlepszą strategię przeszukiwania dla tego problemu i zapisz dla niej statystyki.

Najlepszą strategią jest użyć algorytmów dwukierunkowych takich jak BFS czy Dijkstra (jak widać w tabelce są one najszybsze). Jeżeli jednak czas nie gra dla nas takiej roli i interesuje nas bardziej dokładność rozwiązania to można rozważyć użycie algorytmów jednokierunkowych.

## Sekcja "5 Problem n-puzzli"

1. Graf reprezentujący puzzle 5x5 zajmie w pamięci:  
 $32\text{bits} * 25! = 4.96358721\text{e}26$  - jak widzimy jest to ogromna liczba.
2. Czy graf musi być w całości wygenerowany przed przystąpieniem do przeszukiwania? **NIE**, ponieważ dla konkretnego z wierzchołków iteracyjnie wyznaczamy kolejne możliwe kombinacje ułożenia.
3. Statystyki dla różnych algorytmów i heurystyk dla domyślnego problemu:

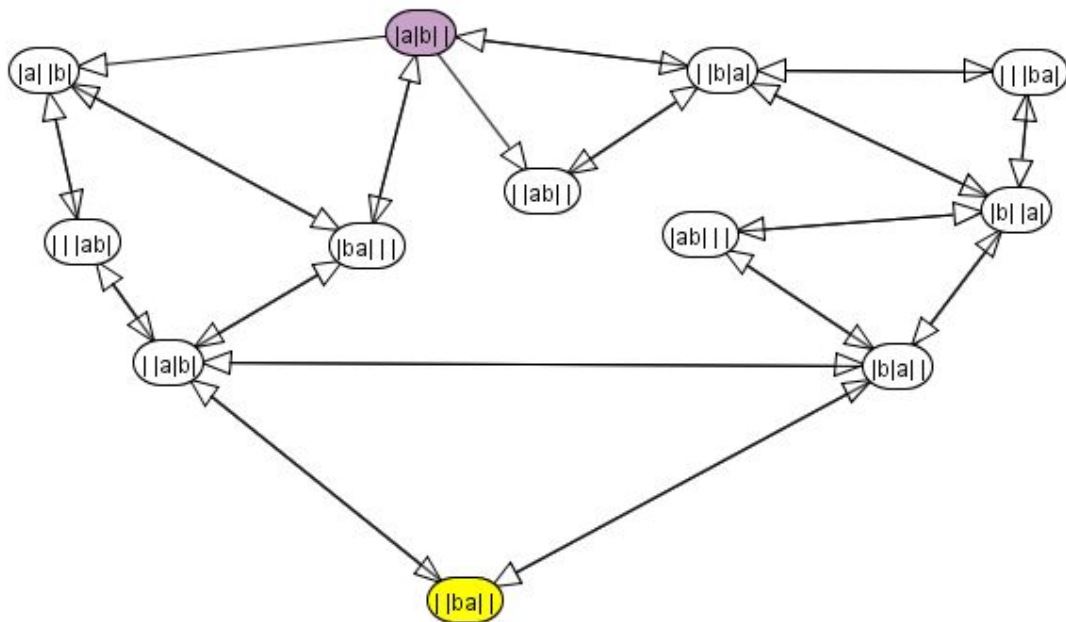
<i>Ustawienia</i>	<i>Długość ścieżki</i>	<i>Open list</i>	<i>Closed list</i>
<i>Breadth-First-Search</i>	6	60	84
<i>Iterative Deepening Search</i>	7	2	17
<i>Greedy (Euclidean)</i>	6	4	7
<i>Greedy (Manhattan)</i>	6	4	7
<i>Greedy (Tile out-of-place)</i>	6	4	7
<i>A* (Euclidean)</i>	6	6	8
<i>A* (Manhattan)</i>	6	4	7
<i>A* (Tile out-of-place)</i>	6	4	7

4. Jak na proces rozwiązywania wpływa zamiana 1 i 2 w stanie początkowym? Jeżeli zmiana jest zauważalna, co jest jej przyczyną?

Liczba kroków zwiększa się znacząco. Przyczyną jest to, że problem jest typu NP-hard i liczenie tego wszystkiego zajmuje bardzo dużo czasu.

## Sekcja „6 Świat klocków”

1. Poniżej znajduje się screenshot z narysowanym grafem stanów dla świata klocków:



2. Wynik rozwiązania dla tego grafu (metoda - Best-First Search):

a. Ścieżka:

- i. |a|b| |
- ii. |a| |b|
- iii. ||a|b|
- iv. ||ba| | (**koniec**)

b. Długość (koszt ścieżki): 3.0

c. Liczba rozwiniętych węzłów: 6

## Sekcja „7 Zagadnienie automatycznego planowania”

### 1. Reguły dla brakujących akcji w solverze STRIPS:

#### a. ACTION II:

```
26 %TODO: ACTION II
27 strips_rule(pick_from(X,Y),
28             [on(X,Y), handempty],
29             [holding(X), clear(Y)],
30             [on(X,Y), clear(X), handempty]).
```

#### b. ACTION III:


```
31 %TODO: ACTION III
32 strips_rule(place(X),
33             [holding(X)],
34             [ontable(X), clear(X), handempty],
35             [holding(X)]).
```

#### c. ACTION IV:

```
36 %TODO: ACTION IV
37 strips_rule(pickup(X),
38             [ontable(X), handempty],
39             [holding(X)],
40             [clear(X), ontable(X), handempty]).
```


### 2. Plany znalezione przez solver dla problemów:

#### a. problem1:

```
 problem1.
Init: [ontable(a), ontable(b), ontable(c), clear(a), clear(b), clear(c), handempty]
Goal: [on(b, c), on(a, b)]
Plan: pickup(b)
      stack(b, c)
      pickup(a)
      stack(a, b)
```

b. problem2:


```

 problem2.
Init: [on(a, c), ontable(b), ontable(c), clear(a), clear(b), handempty]
Goal: [on(b, c), on(a, b)]
Plan: pickup(c)
      stack(c, a)
      pickup(b)
      stack(b, c)
      pick_from(a, c)
      stack(a, b)

```

c. problem3:

```

 problem3.
Init: [on(c, a), ontable(a), ontable(b), clear(b), clear(c), handempty]
Goal: [on(b, c), on(a, b)]
Plan: pickup(b)
      stack(b, c)
      pickup(a)
      stack(a, b)

```

3. Dodatkowy problem:

```

63 %TODO: EXAMPLE IV
64 problem4 :- plan(
65     [on(a,b), on(c, d), ontable(d), ontable(b), clear(a), clear(c), handempty],
66     [on(c,b)]).
67


```

a. stan początkowy:

b. stan końcowy:

c. znaleziony plan:

```

 problem4.
Init: [on(a, b), on(c, d), ontable(d), ontable(b), clear(a), clear(c), handempty]
Goal: [on(c, b)]
Plan: pick_from(c, d)
      place(c)
      pickup(b)
      stack(b, d)
      pickup(c)
      stack(c, b)

```

4. Opisz poniżej, jaką wartość dostrzegasz w rozwiązywaniu problemów bez warunków clear/1

Brak użycia clear/1 powoduje, że problem staje się łatwiejszy. Takie uproszczenie pozwala na przybliżanie się do wyniku (poprzez znajdowanie rozwiązań przybliżających do optymalnego rozwiązania) i finalnie poprawnego wyniku.