Politechnika Warszawska

OKNO

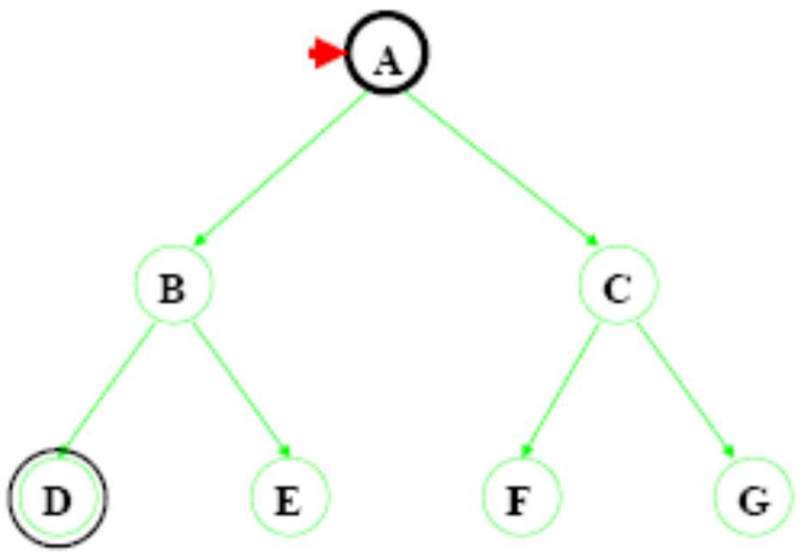
MSI

C3. Przeszukiwanie przestrzeni stanów

Włodzimierz Kasprzak

Zad. 1. Przeszukiwanie wszerz

Wyjaśnić strategię **przeszukiwania wszerz** jako odmiany przeszukiwania **ślepego (niepoinformowanego)** stosowanego do rozwiązywania problemów. **Zilustrować** strategię **przeszukiwania wszerz** dla podanego obok drzewa decyzyjnego, podając kolejność rozwijanych węzłów.

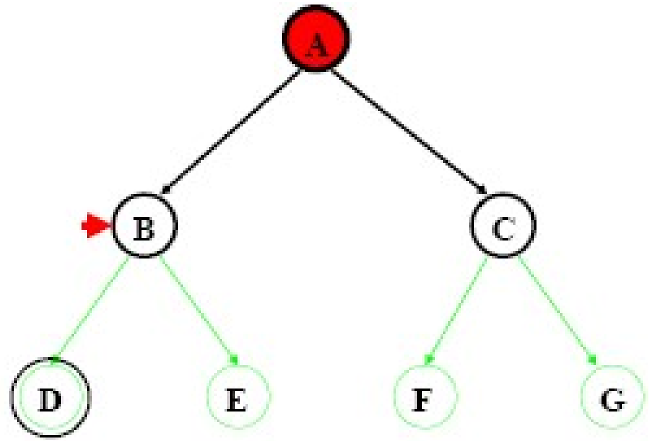


Rozwiązanie

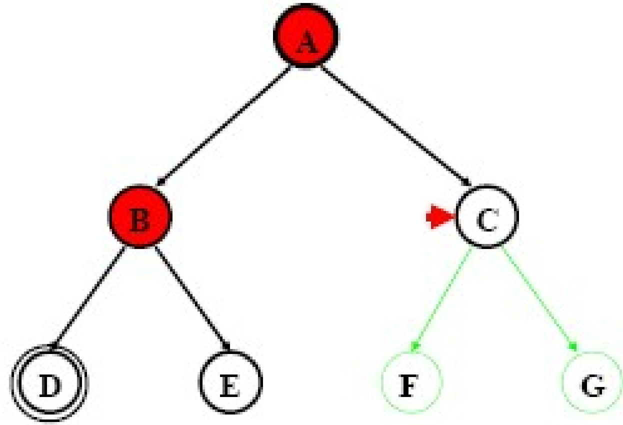
Rozwiązanie 1. Przeszukiwanie wszerz

Iteracja 1: Skraj(1) = {A}, wybierany jest węzeł A, generowane są jego następniki B, C.

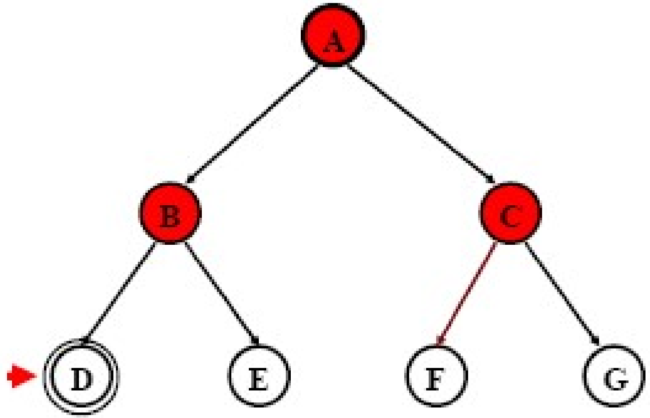
Iteracja 2: Skraj(2) = {B, C}, wybierany jest węzeł B, generowane są jego następniki D i E.



Iteracja 3: Skraj(3) = {C, D, E}, wybierany jest węzeł C, generowane są jego następniki F, G



Iteracja 4: Skraj(4) = { D,E,F,G}, wybierany jest węzeł D, nie są generowane żadne następniki, jeśli D spełnia warunek stopu to KONIEC.



Podsumowanie: rozwijane węzły A, B, C, D.

Zad. 2. Własności przeszukiwania wszerz

Określić własności strategii przeszukiwania wszerz: zupełność, oczekiwana złożoność czasowa (obliczeniowa) i pamięciowa, optymalność.

Rozwiązanie 2

- Zupełność: tak, jeżeli stopień rozgałęzienia b jest skończony.
- Złożoność czasowa: proporcjonalna do liczby rozwijanych węzłów, szacowanej jako: $1+b+b^2+b^3+\ldots+\frac{1}{2}\,b^d = O(b^{d+1})$,

  gdzie b – stopień rozgałęzienia, d – długość ścieżki rozwiązania.
- Wymagania na pamięć: każdy węzeł jest zachowywany w pamięci, $O(b^{d+1})$.
- Optymalność: tak, jeśli koszt każdego kroku jest ten sam, gdyż ten algorytm znajduje najkrótszą ścieżkę.

Zad. 3. Przeszukiwanie z jednolitą funkcją kosztu („*uniform-cost*” *search*)

Określić własności strategii „z jednolitą funkcja kosztu”: zupełność, oczekiwana złożoność czasowa (obliczeniowa) i pamięciowa, optymalność.

Rozwiązanie 3.

<u>Własności</u>

- Zupełność: tak, jeżeli koszt każdej akcji ≥ ε, gdzie $\varepsilon \geq 0$.
- Czas: liczba węzłów o wartości kosztu $g(n)$ ≤ koszt optymalnego rozwiązania, $O(b^{(C^*/\varepsilon)})$, gdzie C^* jest kosztem optymalnego rozwiązania.
- Pamięć: liczba węzłów o koszcie $g(n)$ ≤ koszt optymalnego rozwiązania: $O(b^{(C^*/\varepsilon)})$.
- Optymalność: tak, w sensie minimalizacji kosztu, gdyż węzły rozwijane są zawsze w kolejności zwiększającego się kosztu $g(n)$.

Zad. 4. Własności przeszukiwania w głąb

Określić własności strategii przeszukiwania w głąb: zupełność, oczekiwana złożoność czasowa (obliczeniowa) i pamięciowa, optymalność.

Rozwiązanie 4.

- Zupełność. Tak, jeżeli stopień rozgałęzienia b i maksymalna głębokość drzewa m są wartościami skończonymi (skończona przestrzeń).
- Czas. $O(\frac{1}{2} b^{m+1})$, gdzie m – maksymalna głębokość drzewa. W porównaniu do przeszukiwania wszerz przeszukiwanie w głąb powinno rozwijać znacznie mniej węzłów, gdy $m \cong d$, ale gdy $m >> d$, też prowadzi do dużej liczby węzłów w praktyce. Jednak jeżeli rozwiązania są gęste (istnieje wiele ścieżek z rozwiązaniami) to nawet wtedy (tzn. przy $m >> d$) ta strategia powinna być dużo szybsza niż przeszukiwanie wszerz.
- Pamięć. $O(bm)$ – jedynie liniowa złożoność pamięciowa względem stopnia rozgałęzienia i maksymalnej głębokości rozwiązania. Wystarczy bowiem pamiętanie jedynie aktualnej ścieżki (o maksymalnej długości *m*) i maksymalnie *b* węzłów na każdym poziomie drzewa.

Rozw. 4. Własności przeszukiwania w głąb

• Optymalność.

Nie, gdyż pierwsze znalezione rozwiązanie, na którym kończy się przeszukiwanie, nie musi być wcale najlepsze. Nawet, gdy założymy jednakowy koszt każdej akcji, to znalezione rozwiązanie może być na dłuższej ścieżce niż ewentualnie inne pominięte rozwiązania.

Zad. 5. Własności iteracyjnego pogłębiania (IDS)

Określić własności strategii iteracyjnego pogłębiania: zupełność, oczekiwana złożoność czasowa (obliczeniowa) i pamięciowa, optymalność. Porównać oczekiwane liczby węzłów generowanych w DLS i IDS.

Rozwiązanie 5.

• Oczekiwana liczba węzłów generowanych do głębokości d przy stopniu rozgałęzienia b w przeszukiwaniu z ograniczoną głębokością (DLS):

$$N_{\mathrm{DLS}} = b^0 + b^1 + \tfrac{1}{2}\, b^2 + \ldots + \tfrac{1}{2}\, b^{d\text{-}2} + \tfrac{1}{2}\, b^{d\text{-}1} + \tfrac{1}{2}\, b^d$$

• Oczekiwana liczba węzłów generowanych do głębokości d przy stopniu rozgałęzienia b w przeszukiwaniu z iteracyjnym pogłębianiem (IDS):

$$N_{\mathrm{IDS}} = (d+1) \cdot b^0 + d \cdot b^1 + (d\text{-}1) \cdot b^2 + \ldots + 3 \cdot b^{d\text{-}2} + 2 \cdot b^{\mathrm{d-1}} + 1 \cdot \tfrac{1}{2}\ b^{\mathrm{d}}$$

Np. dla $b = 10, d = 5$: $N_{\mathrm{DLS}} = 1 + 10 + 50 + 500 + 5.000 + 50.000 = 55.561$

$N_{\mathrm{IDS}} = 6 + 50 + 400 + 3.000 + 20.000 + 50.000 = 73{,}456$

W tym przypadku nadmiar liczby węzłów dla IDS w porównaniu do DLS wynosi: $(73.456 – 55.561) / 55.561 \cong 0.322 \rightarrow 32.2\ \%$.

Dla większych wartości b , d ten nadmiar procentowy maleje.

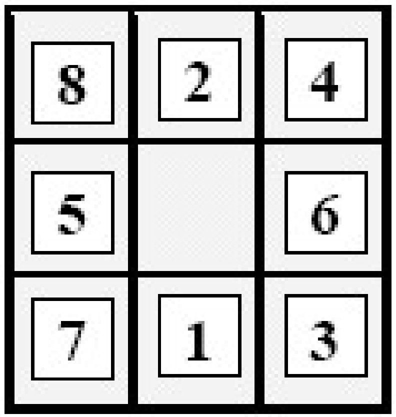
Rozw. 5. Własności iteracyjnego pogłębiania

- Zupełność? Tak, jeżeli stopień rozgałęzienia b jest wartością skończoną.
- Czas? $(d+1)\, b^0 + d\, b^1 + (d-1)\, b^2 + \ldots + 1 \cdot ½\ b^d = O(\, b^{d+1}\,)$
- Pamięć? $O(bd)$, co wynika ze stosowania przeszukiwania w głąb dla każdego ograniczenia l.
- Optymalność? Tak, jeżeli koszt każdej akcji jest równy. Znajdujemy tu bowiem zawsze najkrótsze rozwiązanie.

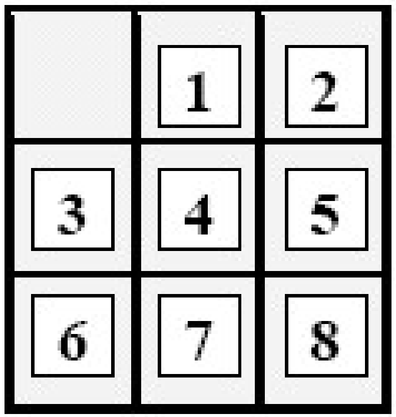
Zad. 6. Przeszukiwanie z heurystyką - strategia zachłanna

W problemie „8 puzzli” znane są: stan początkowy i wymagany stan końcowy, jak na rysunku.

Rozwiązać problem złagodzony wobec problemu 8 puzzli, w którym pojedyncza akcja polega na natychmiastowym przemieszczeniu (jedną akcją) jednego kafelka na wymagane miejsce docelowe (można kłaść jeden na drugi), stosując strategię **zachłanną przeszukiwania poinformowanego** przy zastosowaniu heurystyki odpowiadającej **metryce Manhattan**.



Stan początkowy



Stan końcowy

Rozwiązanie 6. Strategia zachłanna dla złagodzonego problemu 8 puzzli (1)

Rozwiązanie (wskazówka)

Mamy do czynienia z problemem, w którym „jeden stan problemu = jeden węzeł drzewa przeszukiwania”.

Przypomnijmy też, że strategia zachłanna kieruje się jedynie częścią heurystyczną w ocenie węzłów: $f(n) = h(n)$.

W zadanym problemie dla stanu początkowego: $h(Start) = 4 + 1 + 2 + 2 +3+1+2+3 = 18$.

Oczywiście dla stanu docelowego: $h(cel)=0$.

Inicjalizujemy drzewo przeszukiwania węzłem startowym $n0$ o ocenie $f(n0) = h(n0) = 18$.

Krok 1:

W pierwszej iteracji (kroku) przeszukiwania skraj drzewa zawiera jedynie węzeł $n0$: $Skraj(1)= \{n0\}$.

Rozwiązanie 6. Strategia zachłanna dla złagodzonego problemu 8 puzzli (2)

Wybieramy węzeł $n0$ i generujemy 8 możliwych następników: $n1, n2, ..., n8$. Reprezentują one przewidywane konfiguracje (kafelków) osiągane po wykonaniu akcji przeniesienia odpowiednio kafelka: 1, 2, ..., 8; na swoje docelowe miejsce. Oceniamy te węzły i porządkujemy od najmniejszego do największego kosztu resztkowego. „Największy uzysk" daje akcja przeniesienia kafelka 8: $f(n8) = h(n8) = 14$. „Najgorszą akcję" reprezentuje wybór n2 lub n7, gdyż ich oceny wynoszą 17.

Krok 2:

W kroku 2 wybieramy więc do rozwinięcia węzeł $n8$ i generujemy 7 możliwych jego następników (kafelek 8 jest już na swoim docelowym miejscu i zakładamy, że akcja jego przeniesienia w inne miejsce nie istnieje).

Itd. kontynuujemy aż do kroku 8, w którym osiągniemy stan docelowy.

Zad. 7. Przeszukiwanie A*

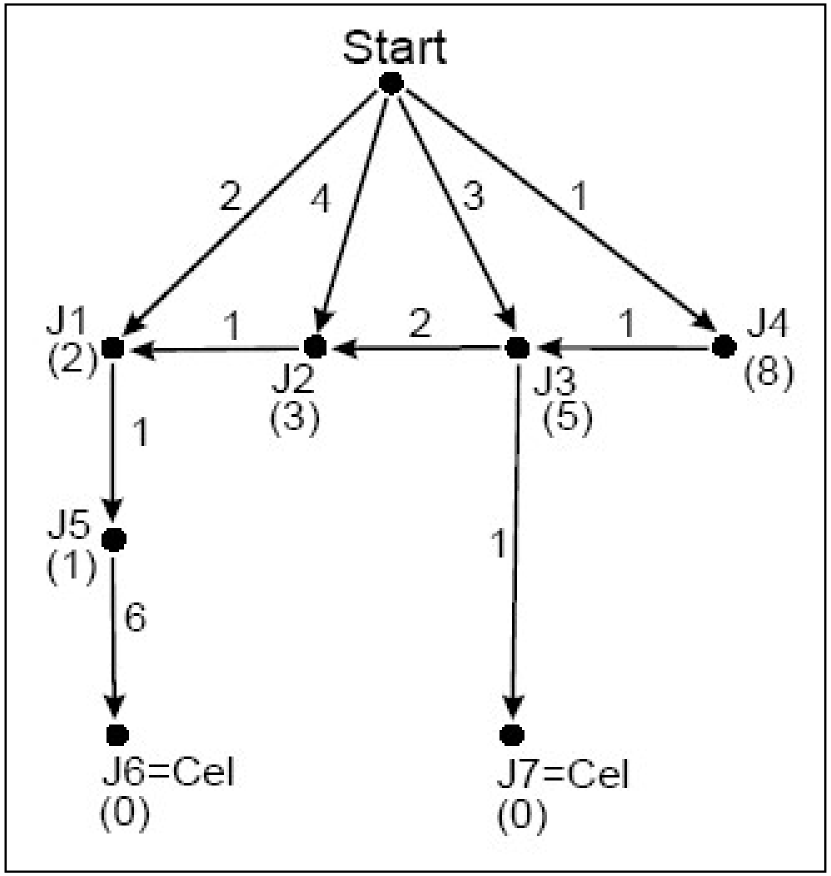
W podanym niżej grafie pewnego problemu koszty operacji pomiędzy stanami podane są przez liczby przy łukach, a oszacowanie kosztów pozostałych do celu (heurystyka) - liczby w nawiasach przy węzłach.

A) Zilustrować działanie strategii **przeszukiwanie A*** dla zadanego grafu problemu.

B) Czy heurystyka jest **dopuszczalna**?

C) Czy znaleziono **optymalne** rozwiązanie?

Uwaga: dwa stany spełniają warunek stopu (są docelowe), jednak tylko jeden z nich reprezentuje optymalne rozwiązanie.



Rozw. 7. Przeszukiwanie A*

A)

Krok *i*	**OPEN(*i*), f(n)**	**CLOSED(*i*)**	**Wybór węzła → generowane następniki**
1	{Start} f(Start) = 0	{}	Start → n1 (dla J1), n2 (dla J2), n3 (dla J3), n4 (dla J4)
2	{n1,n2,n3,n4} f(n)={4,7,8,9}	{Start}	n1 → n5 (dla J5)
3	{n5,n2,n3,n4} f(n)={4,7,8,9}	{Start,n1}	n5 → n6 (dla J6)
4	{n2,n3,n4,n6} f(n)={7,8,9,9}	{Start,n1,n5}	n2 → n1' , ale jest gorszy od już istniejącego n1, tzn. g(n1') > g(n1)
5	{n3,n4,n6} f(n)={8,9,9}	{Start,n1,n5,n2}	n3 → n2', ale jest gorszy od już istniejącego n2, tzn. g(n2') > g(n2) n3 → n7 (dla J7),
6	{n7,n4,n6} f(n)={4,9,9}	{Start,n1,n5,n2,n3}	n7, Stop(J7) = true, stan J7 spełnia warunek stopu → **koniec**
			Rozwiązanie: Start → J3 → J7

Rozw. 7. Przeszukiwanie A*

B) W podanym problemie heurystyka **nie jest dopuszczalna**. Oszacowania kosztów resztkowych dla węzłów J3 i J4 nie są optymistyczne (sa wyższe od rzeczywistych kosztów akcji pozostałych jeszcze do wykonania).

C) Znalezione rozwiązanie nie jest optymalne – ścieżka:

„Start → J3 → J7” ma koszt 4.

Optymalna ścieżka to: „Start→J4→J3→J7” o koszcie 3.

Zadanie 8. Porównanie strategii przeszukiwania poinformowanego

Rozwiązać problem znalezienia optymalnej ścieżki (od S do celu) stosując poinformowane strategie przeszukiwania grafu:

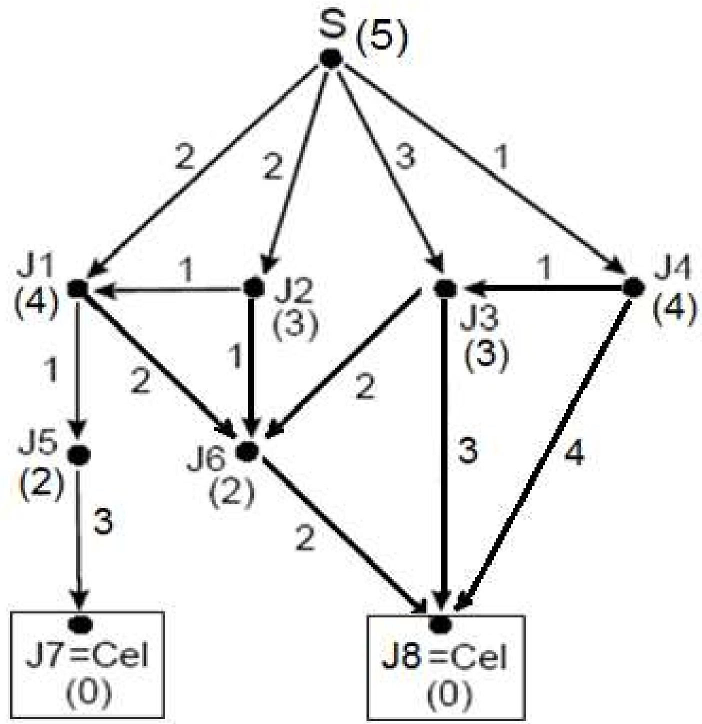
A) Strategię zachłanną, B) Przeszukiwanie A*.

W nawiasach przy węzłach podano heurystykę (oszacowanie kosztów resztkowych) dla funkcji kosztu danego węzła, a przy łukach – koszty akcji przejścia pomiędzy węzłami.

W każdym z przypadków A) i B) podać i wyjaśnić: rozwijane drzewo decyzyjne – kolejność wizytowanych węzłów; zwracane rozwiązanie (ścieżkę i jej koszt).

C) Sprawdzić czy heurystyka jest dopuszczalna.

D) Sprawdzić czy heurystyka jest spójna.



Rozwiązanie 8 (1)

A) Strategia zachłanna

- Funkcja oceny (koszt): **f(n) = h*(n)** (oszacowanie kosztów resztkowych)
- Horyzont (skraj drzewa przeszukiwania) **globalny**.

Krok	n ← wybrany węzeł	Test końcowy (n)= ?	Następniki	CLOSED	OPEN	Uwagi
Init				{ }	{S}	
1	S, f(S)= 5	NIE	J1, J2, J3, J4	{S}	[J2, J3, J1, J4] [3, 3, 4, 4]	Kolejność J2, J3 zależy od dodatkowej reguły
2	J2,f(J2)=3	NIE	J1’, J6	{S, J2}	[J6, J3, J1, J4] [2, 3, 4, 4]	J1’ nie zostaje dodany.
3	J6, f(J6)=2	NIE	J8	{S, J2, J6}	[J8, J3, J1, J4] [0, 3, 4, 4]	
4	J8, f(J8)=0	TAK	→STOP			

Rozwiązanie 8 (2)

- Ścieżka: **S→ J2 → J6 → J8**
- Koszt: 2 + 1 + 2 = **5**.
- Liczba wizytowanych (wybieranych) węzłów: 4

Uwaga: w strategii przeszukiwania zachłannego unika się wstawiania kolejnego generowanego węzła jeśli jest równoważny z już istniejącym w OPEN lub CLOSED.

B) Strategia A*

- Funkcja oceny (koszt): **f(n) = g(n) + h*(n)**
- Horyzont (skraj drzewa przeszukiwania) **globalny**.

Rozwiązanie 8 (3)

Krok	n ← wybrany węzeł	Test końcowy (n)	Następniki	CLOSED	OPEN	Uwagi
Init				{}	{ S }	
1	S, f(S) = 5	NIE	J1, J2, J3, J4	{S}	[J2, J4, J1, J3] [5, 5, 6, 6]	Może też być [J4, J2,] i [J3, J1]
2	J2, f(J2) =5	NIE	J1', J6	{S, J2}	[J4, J6, J1, J3] [5, 5, 6, 6]	J1' nie będzie dodany
3	J4, f(J4) =5	NIE	J3', J8	{S, J2, J4}	[J6, J3', J8, J1] [5, 5, 5, 6]	J3'(f=5) zastąpi J3(f=6)
4	J6, f(J6)= 5	NIE	J8'	{S, J2, J4, J6}	[J3', J8, J1] [5, 5, 6]	J8' nie zostanie dodany
5	J3', f(J3')= 5	NIE	J6', J8' [6, 5]	{S, J2, J4, J6, J3'}	[J8, J1] [5, 6]	J6' ani J8' nie zostaną dodane
6	J8, f(J8)= 5	TAK	→ STOP			

Rozwiązanie 8 (4)

- Ścieżka: S→ J4 → J8
- Koszt: 5.
- Ścieżka optymalna – nie ma lepszej ścieżki rozwiązania, co najwyżej są inne o tym samym koszcie.
- Liczba wizytowanych (wybieranych) węzłów: 6

Uwaga: w strategii A* sprawdza się nowo wygenerowany węzeł przed jego dodaniem zarówno ze zbiorem CLOSED jak i OPEN.

C) Sprawdzenie, czy heurystyka jest DOPUSZCZALNA.

Dla każdego stanu N sprawdzamy warunek: czy rzeczywisty minimalny koszt resztkowy [min h(N->cel)] jest niemniejszy od jego oszacowania h*(N).

Rozwiązanie 8 (5)

Heurystyka jest dopuszczalna – dla wszystkich stanów podany warunek jest spełniony:

Stan N	1: h*(N)	2: h(N->cel)	1 <= 2 ?	Wszystkie Tak?
S	5	5	Tak	
J1	4	4	Tak	
J2	3	3	Tak	
J3	3	3	Tak	
J4	4	4	Tak	
J5	2	3	Tak	
J6	2	2	Tak	
J7, J8	0	0	Tak	**Tak**

Rozwiązanie 8 (6)

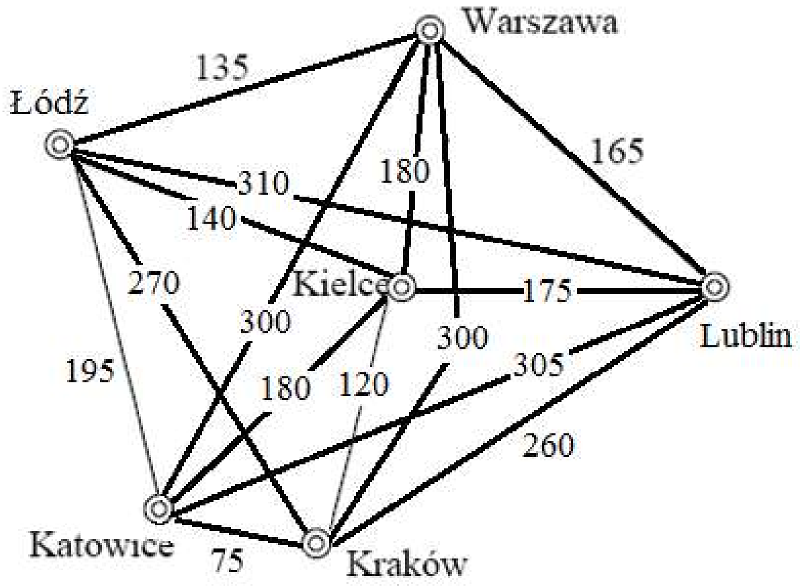
D) Sprawdzanie, czy heurystyka jest SPÓJNA.

- Dla każdego łuku C(n->m) sprawdzamy, czy spełniony jest warunek „trójkąta": czy zachodzi [h*(n) <= C + h*(m)] ? tzn. czy f(n) <= f(m) ?

C (n->m)	1: h*(n)	2: C+ h*(m)	1 <= 2 ?	Wszystkie Tak?
S-> J1	5	6	Tak	
S->J2	5	5	Tak	
S->J3	5	6	Tak	
S->J4	5	5	Tak	
J1->J5	4	3	Nie	
J1->J6	4	4	Tak	
J2->J1	3	5	Tak	
J2->J6	3	3	Tak	
J3->J6	3	4	Tak	
J3->J8	3	3	Tak	
J4->J3	4	4	Tak	
J4->J8	4	4	Tak	
J5->J7	2	3	Tak	
J6->J8	2	2	Tak	**Nie**

Zad. 9. Przeszukiwanie A* dla rozwiązania problemu komiwojażera

Zastosować przeszukiwanie A* dla rozwiązania konkretnego „problemu komiwojażera” zadanego poniższym rysunkiem:



- Założyć, że ścieżka rozpoczyna i kończy się w Warszawie.
- Zdefiniować nietrywialną heurystykę (tzn. inną niż $h(n) = 0$).
- Pokazać drzewo decyzyjne rozwijane dla znalezienia rozwiązania, podać ścieżkę rozwiązania i jej koszt (w km, jako suma odległości częściowych podanych na rysunku).

Rozwiązanie 9 (1)

- Zwracamy uwagę na to, aby heurystyka była dopuszczalna.
- Wartość heurystyki można związać, jak zawsze dotąd, z miastem „n”, na którym kończy się dana ścieżka częściowa. Jednak będzie to słaba heurystyka. Lepszym rozwiązaniem jest zdefiniowanie „stanu” jako ścieżki częściowej zakończonej danym miastem „n”.

  Np. stan = ścieżka(start-> … -> n).
- W celu budowy konkretnych heurystyk dla „stanów – ścieżek” zdefiniujemy najpierw heurystyki pomocnicze dla poszczególnych miast.
- Heurystyka miast, **h1_miasto(M)** = {koszt połączenia miasta „M” z miastem startowym „W-wa”}:

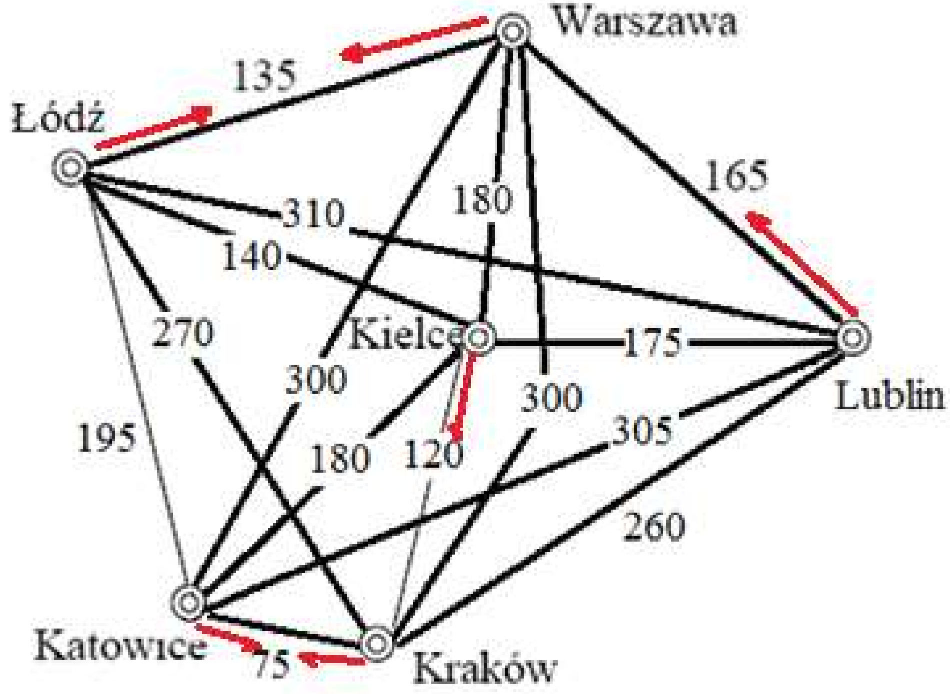
Miasto M	**h1_miasto(M)**
Warszawa	0
Łódź	135
Kielce	180
Katowice	300
Kraków	300
Lublin	165

Rozwiązanie 9 (2)

- Heurystyka miast:

h2_miasto(M) = {minimalny koszt połączenia węzła „n” z jego sąsiadem}

Miasto M	h2_miasto(M)
Warszawa	135
Łódź	135
Kielce	120
Katowice	75
Kraków	75
Lublin	165



Rozwiązanie 9 (3)

- Teraz zdefiniujmy heurystyki dla „stanów-ścieżek”.
- Prosta heurystyka: h1(„ścieżka: start-> …->M”) = [suma minimalnych kosztów połączenia pozostałych jeszcze miast z sąsiadem + koszt minimalny dla miasta „start”].
- Heurystyka h2(„ścieżka: start-> …->M”) = {suma minimalnych kosztów połączenia pozostałych jeszcze miast z sąsiadem jeszcze nieodwiedzanym lub M lub miastem „start” + taki koszt dla miasta „start”}

Z konstrukcji obu heurystyk wynika, że h2 dominuje nad h1 i obie są dopuszczalne.

- Inna prostsza heurystyka: h3(„ścieżka: start-> …-> M) = {(minimalny koszt jednego połączenia w grafie) * (liczba miast pozostałych do wizytowania)}

Przykłady heurystyk:

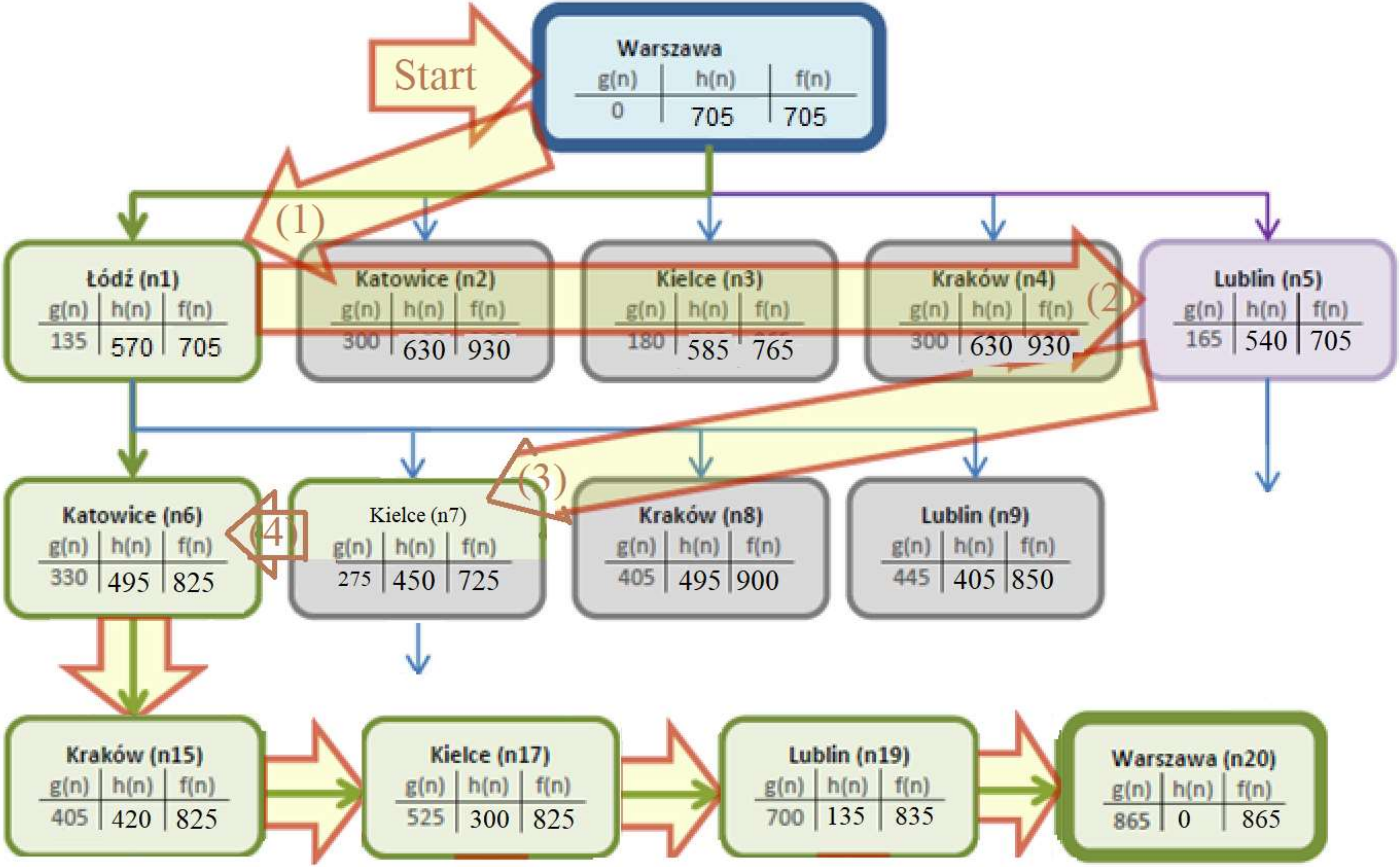
h1(„W-wa -> Łódź -> Katowice”) = (Kraków) 75 + (Kielce) 120 + (Lublin) 165 + Warszawa 135 = 495 (rzeczywisty koszt resztkowy = 535)

h2(„W-wa -> Łódź-> Katowice”) = (Kraków) 75 + (Kielce) 120 + (Lublin) 165 + (Warszawa) 165 = 525

h3(„W-wa -> Łódź-> Katowice”) = 75 * 4 = 300

Rozwiązanie 9 (4)

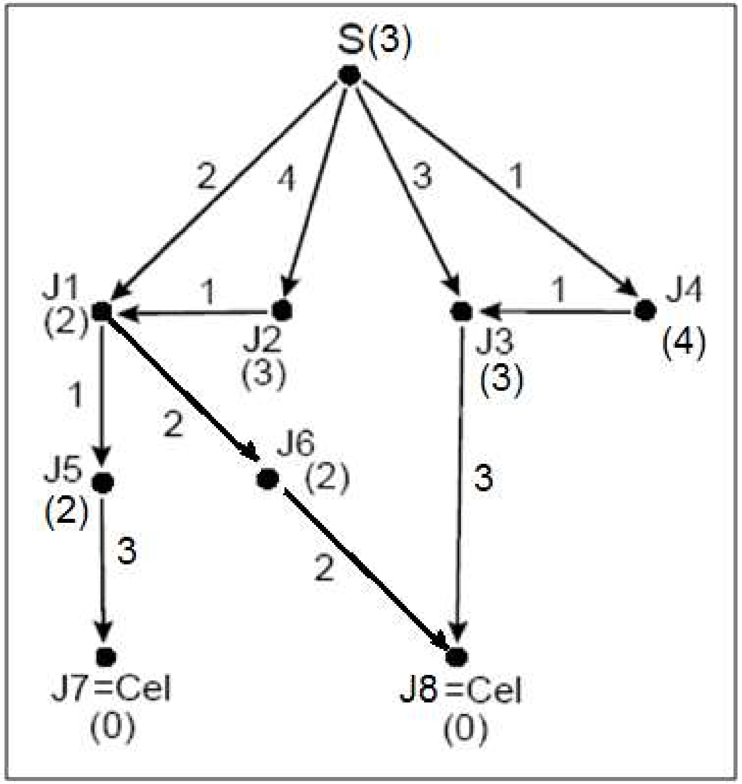
Drzewo decyzyjne (fragment) przy założeniu heurystyki **h1:**



Zad. 10. Porównanie strategii A*-podobnych

Rozwiązać podany problem znalezienia optymalnej ścieżki (od S do celu) stosując cztery A*-podobne strategie przeszukiwania:

A) A*,
B) IDA*,
C) SMA* z limitem pamięci 3 i ścieżki 2,
D) RTA* („z minusem") z podglądem o jedną akcję do przodu.



W nawiasach podano heurystykę (oszacowanie kosztów resztkowych) dla funkcji kosztu ścieżki.

W każdym z przypadków A), B), C) i D) podać: zasadę strategii, rozwijane drzewo decyzyjne i zwracaną ścieżkę.
Porównać ze sobą wyniki i skomentować je.

Rozwiązanie 10.A): A*

A*

Funkcja kosztu: $f(n)=g(n) + h(n)$

Drzewo decyzyjne podobnie budowane jak w rozwiązaniu zad. 7 (ale nie jest to identyczna przestrzeń).

Ścieżka rozwiązania: S → J4 → J3 → J8.
Koszt ścieżki = 5

Uwaga: składowa heurystyczna $h(n)$ jest dopuszczalna →
znaleziona ścieżka jest optymalna

Rozwiązanie 10.B): IDA* (1)

IDA*

Stosuje funkcję kosztu, $f(n)=g(n) + h(n)$, ale jedynie do określania warunków realizacji strategii ślepego przeszukiwania w głąb w kolejnych iteracjach procesu przeszukiwania.

Te warunki to:

- Limit kosztu ścieżki $f_b(I)$ w I-tej iteracji,
- Warunek rozwijania węzła n w I-tej iteracji: gdy $f(n) \leq f_b(I)$

Drzewo decyzyjne → na kolejnych slajdach

Ścieżka rozwiązania: S → J4 → J3 → J8 . Koszt ścieżki = 5

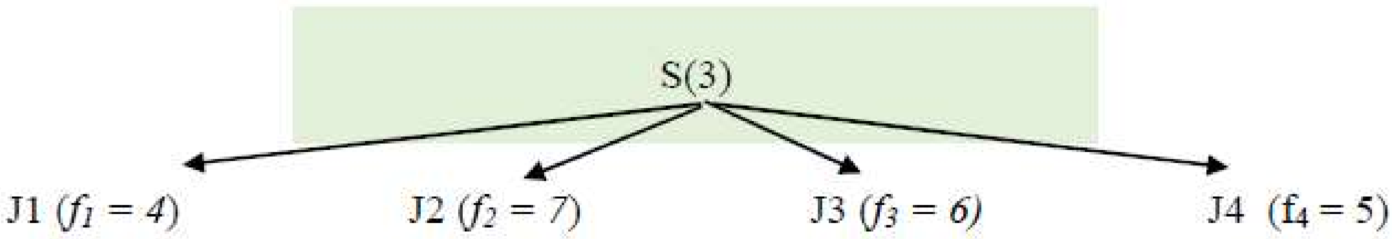
Warunkiem stosowania tego algorytmu jest, aby heurystyka była dopuszczalna i spójna. W tym przypadku obie te własności zachodzą.

Rozwiązanie 10 B): IDA* (2)

<u>IDA* search principle</u>: use of cost function f(n)=g(n) + h(n); an iteration of depth-first search, every depth-first search is limited by current cost threshold f_b(i), only nodes „n" are considered, that satisfy: f_b(i) $\leq$ f(n)

<u>Step-by-step decision tree</u>

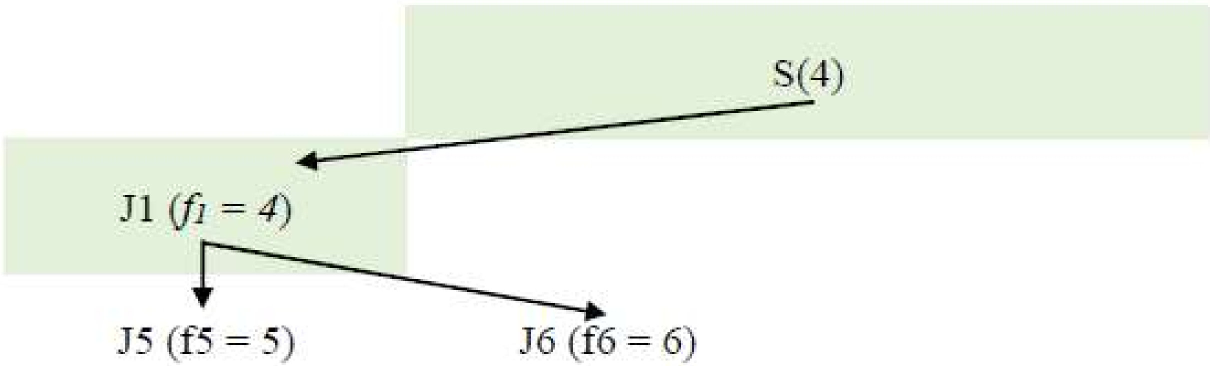
Iteration 1) Cost threshold(1) = f_b(1) = 3 = f(Start)



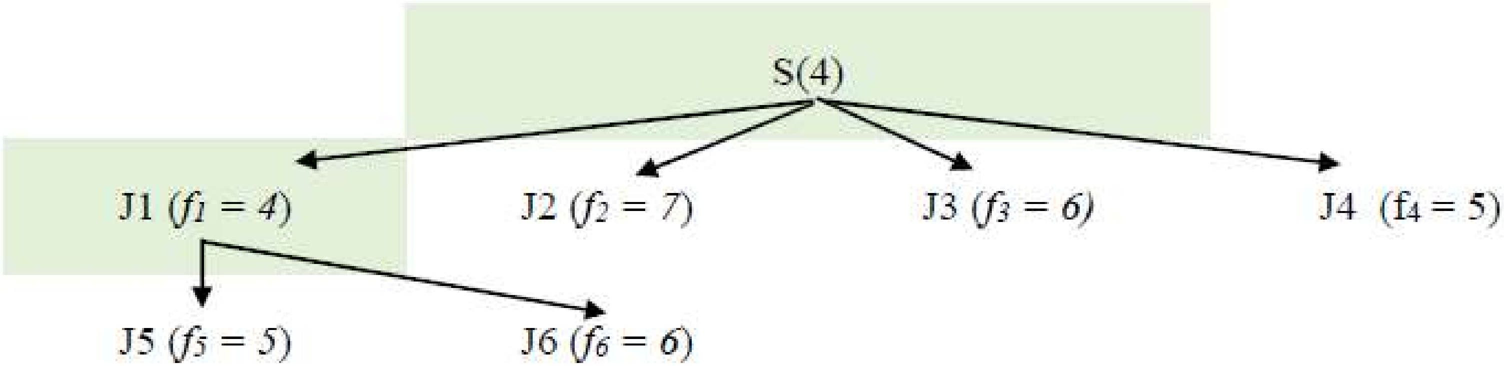
No continuation.

Increase f_b to the smallest value of leaf nodes → f_b(2) = 4

Iteration 2) Cost threshold(1) = f_b(2) = 4

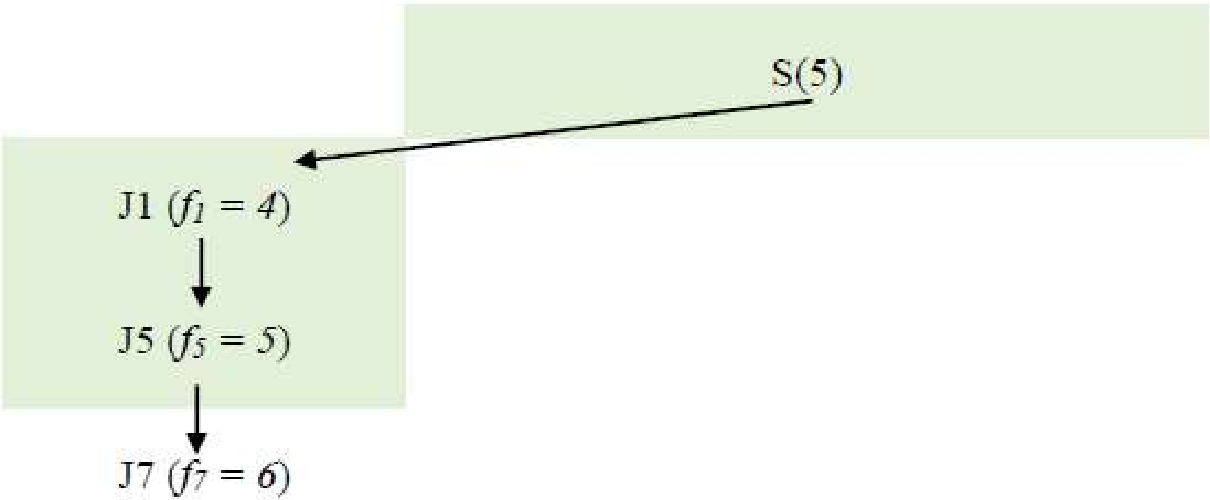


Rozwiązanie 10 B): IDA* (3)

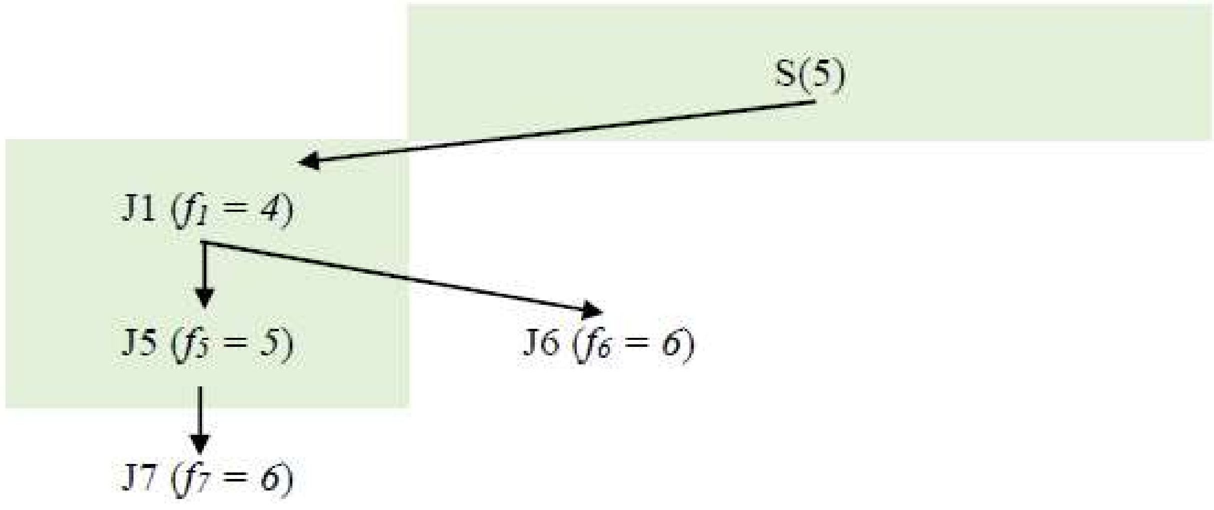


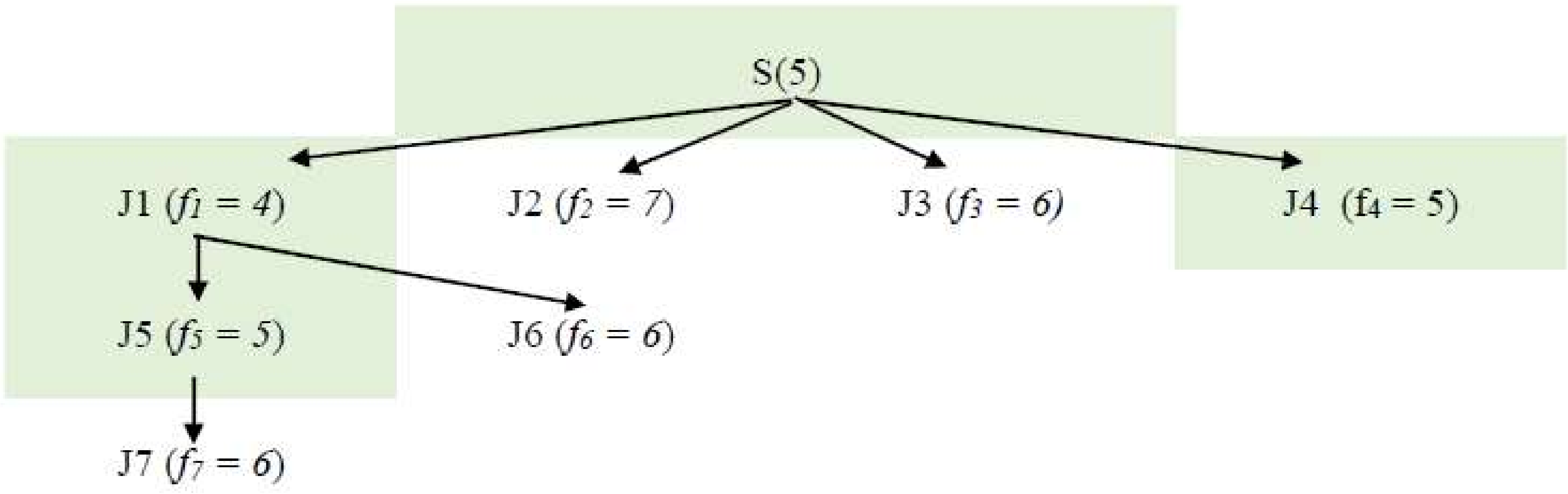
No continuation of iteration (2).
Increase f_b to the smallest value of leaf nodes → f_b(3) = 5

Iteration 3) Cost threshold(1) = f_b(3) = 5

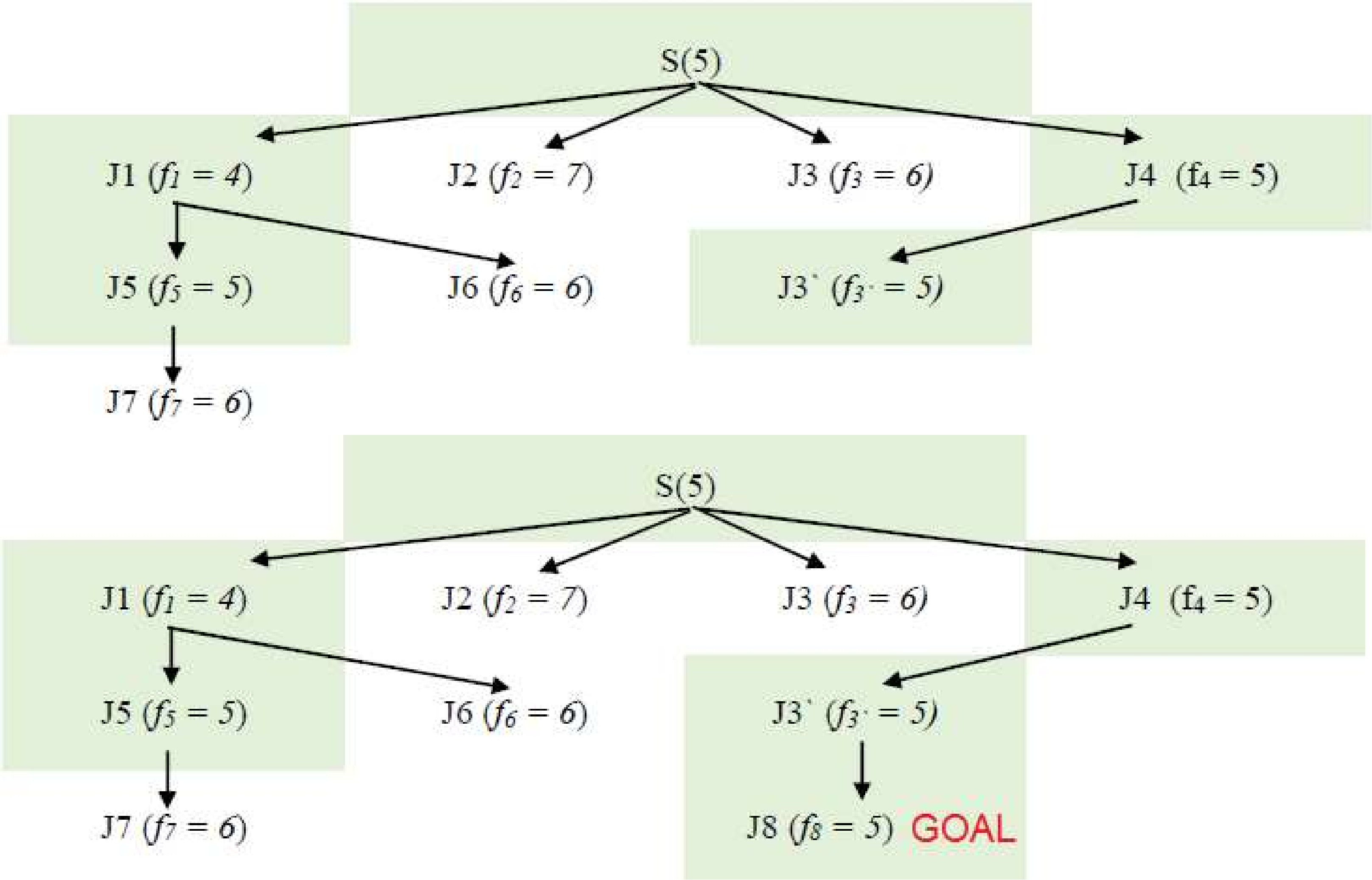


Rozwiązanie 10 B): IDA* (4)





Rozwiązanie 10 B): IDA* (5)



Goal found. Solution path: S → J4 → J3 → J8 . Cost(path) = 5

Comment: heuristics is admissible and consistent → optimal solution path found

Rozwiązanie 10 C): SMA* (1)

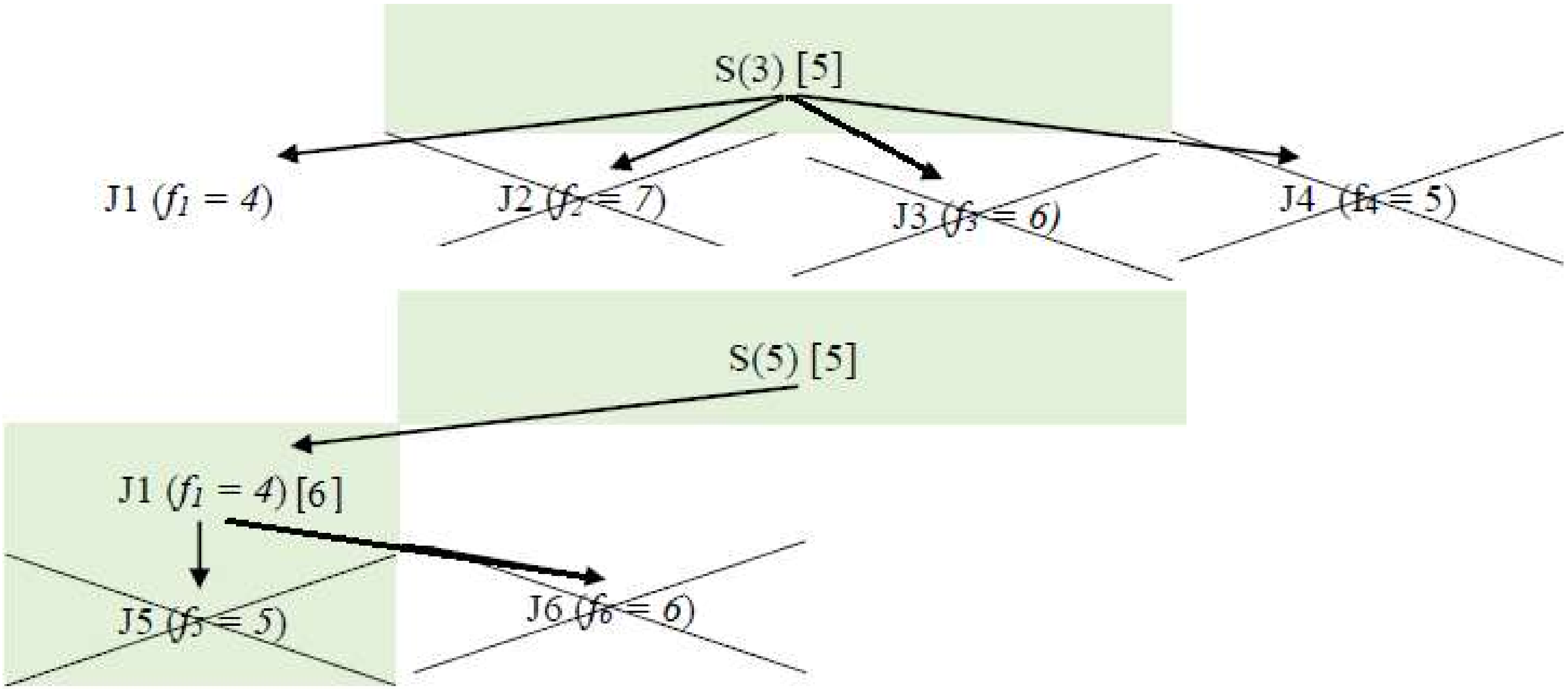
SMA* with memory limit = 3.

Search principle: cost function f(n)=g(n) + h(n); but memory for tree nodes is limited.

Storing the current-best successor node only, while the second-best successor cost is remembered in the parent node.

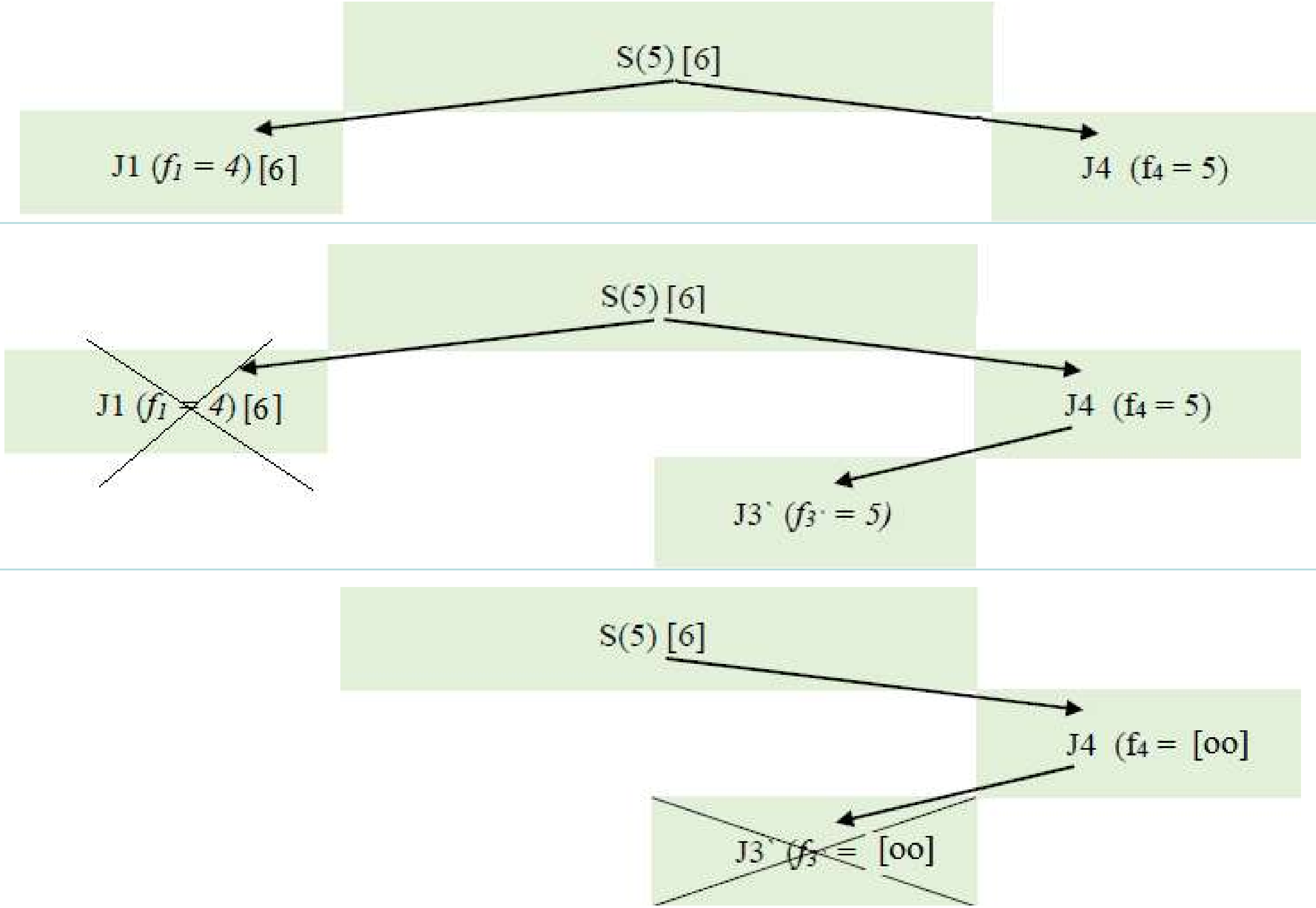
Pruning paths longer than the limit and modifying the parent node cost.

Step-by-step decision tree



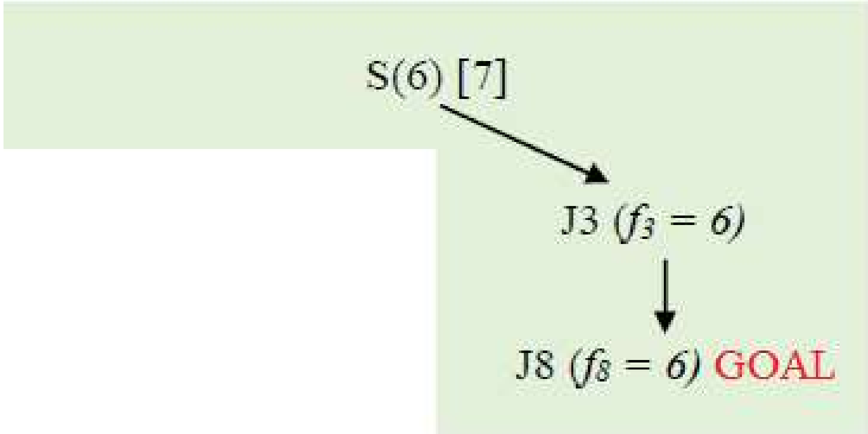
Rozwiązanie 10 C): SMA* (2)

Step-by-step decision tree



Rozwiązanie 10 C): SMA* (3)

<u>Step-by-step decision tree</u>



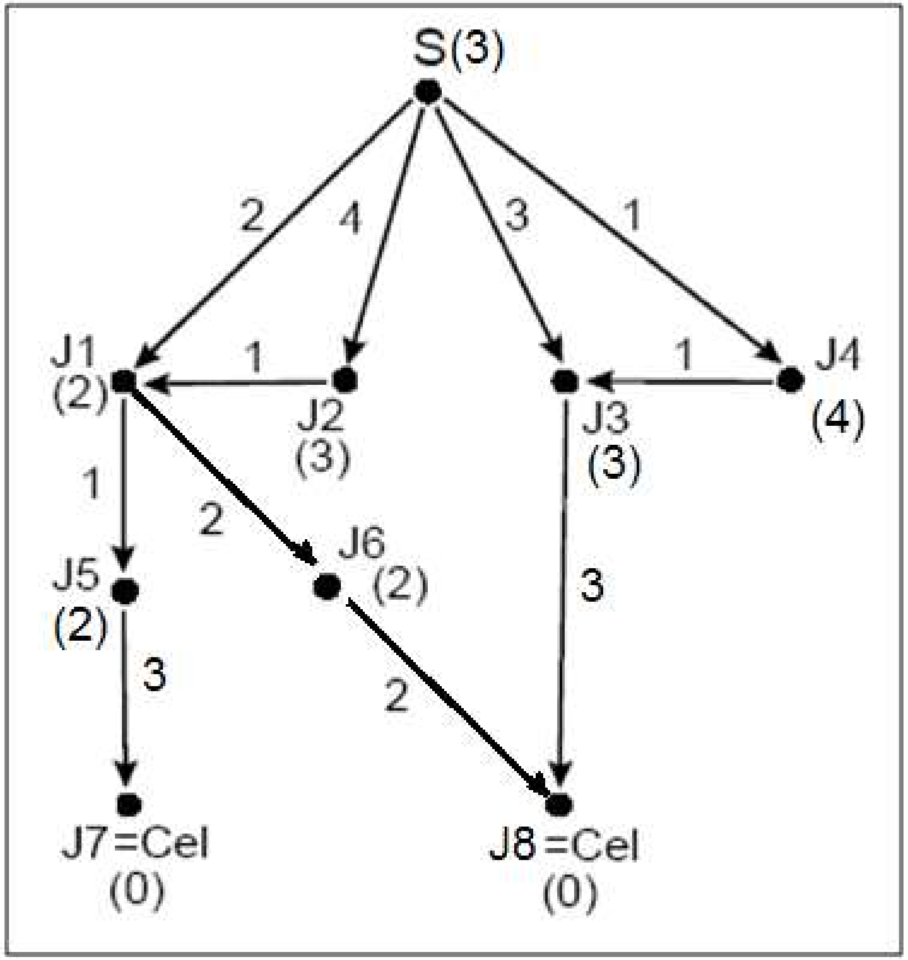
Goal found. Solution path: S → J3 → J8 . Cost(path) = 6

Comment: heuristics is admissible and consistent, but the optimal solution is not found due to memory limit.

For the optimal path, the memory limit should be increased to 4.

Rozwiązanie 10 D) RTA*

Przypomnijmy zadanie: rozwiązać podany problem znalezienia optymalnej ścieżki (od S do celu):



Rozwiązanie 10 D): RTA* (1)

RTA* (wersja z „minusem” dla nawrotu):

1. Aktualny węzeł = węzeł startowy;

   f(Aktualny) = Mini-Min(Aktualny). Wykaz = { }.

2. Jeśli aktualny węzeł jest końcowy to **zakończ.**

3. Wybierz następny węzeł z lokalnego horyzontu bezpośrednich następników aktualnego węzła i węzła-rodzica:

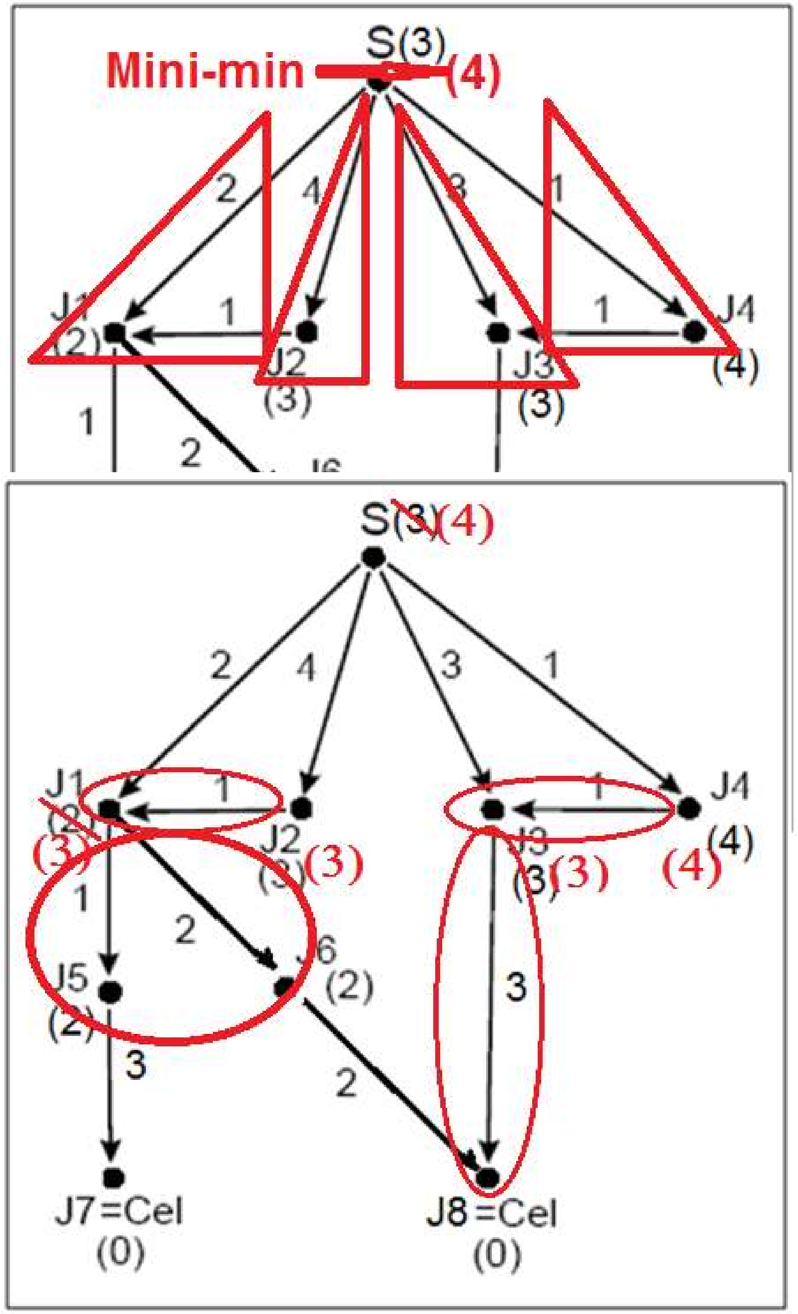
- Dla każdego następnika (nie będącego w Wykazie) „podejrzyj ścieżkę wprzód” – lepiej poinformowana decyzja dzięki przeszukiwaniu „mini-min” z „alfa-przycinaniem” ;
- Wybierz najlepszego następcę (ewentualny wybór węzła-rodzica, jeśli jest korzystniejszy od następników);
- Zastąp ocenę aktualnego węzła przez ocenę „drugiej najlepszej” decyzji (dla uniknięcia pętli).
- Zapisz aktualny węzeł do tablicy.
- Nowy aktualny węzeł = najlepszy następnik

4. Kontynuuj od kroku 2.

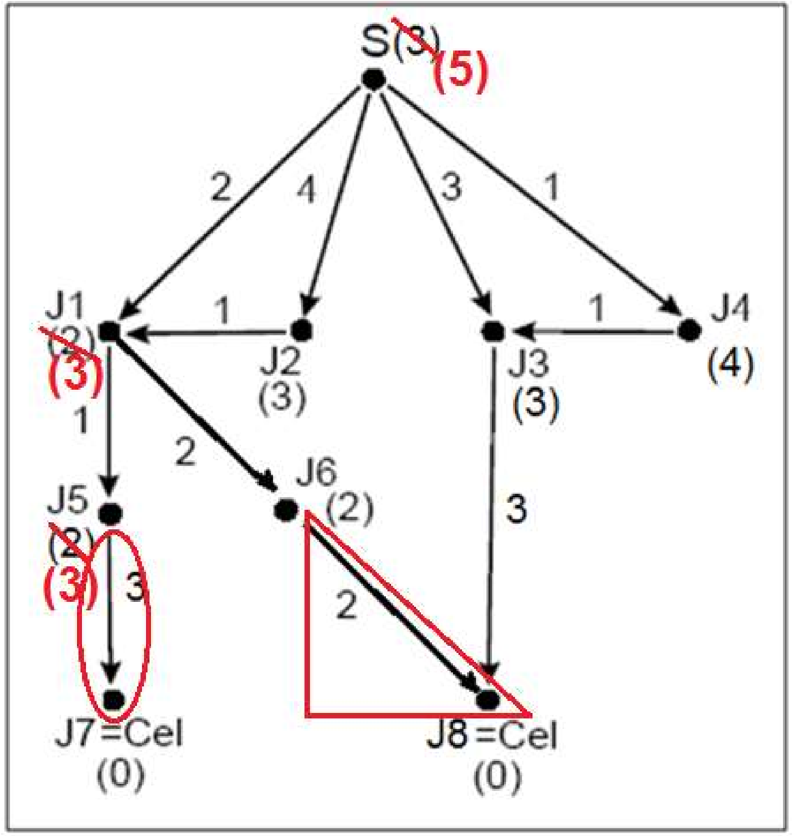
Rozwiązanie 10 D): RTA* (2)

Init: Aktualny= ‚S';
f(‚S') = Minimin(‚S', limit=1) = 4;

<u>Iteracja 1:</u>
TTest(‚S') = False
FOR Następniki (‚S')DO
„Podgląd w przód" →
h(J1) = 3, f(J1) = 2 + 3 = 5 (pierwszy)
h(J2) = 3, f(J2) = 4 + 3 = 7
h(J3) = 3, f(J3) = 3 + 3 = 6,
h(J4) = 4, f(J4) = 1 + 4 = 5 (drugi)
Wykaz(1) = { ‚S', f_{drugi}(S)= 5 }
Aktualny = ‚J1'



Rozwiązanie 10 D): RTA* (3)



<u>Iteracja 2:</u>

TTest(‚J1') = False

FOR Następniki (‚J1')DO

„Podgląd w przód" →

h(J5) = 3, f(J5) = 1 + 3 = 4 (drugi)

h(J6) = 2, f(J6) = 2 + 2 = 4

h(S) = 5, f(S) = - 2 + 5 = 3 (pierwszy)

Wykaz(2) = { ‚J1', f_{drugi}(J1)= 4 }

Aktualny = ‚S' (nawrót)

<u>Iteracja 3:</u>

TTest(‚S') = False

FOR Następniki (‚S')DO

„Podgląd w przód" → nie wykonuj, gdyż ‚S' jest już w Wykazie

h(J1) = 4, f(J1) = 2 + 4 = 6 (drugi)

h(J2) = 3, f(J2) = 3 + 4 = 7

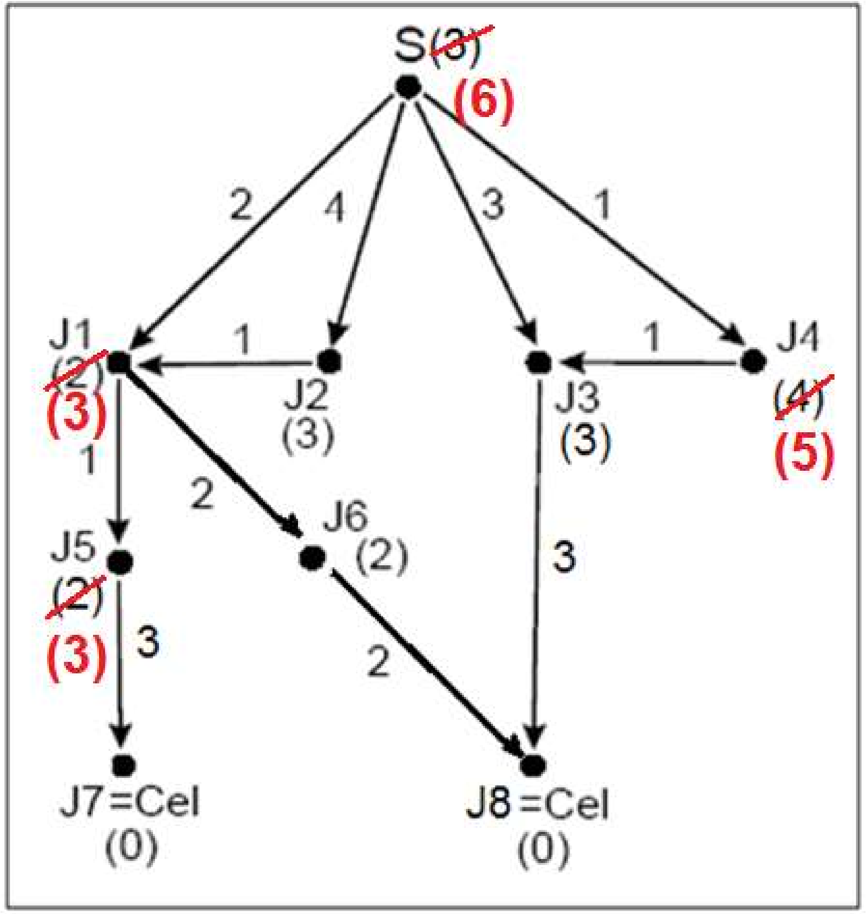
h(J3) = 3, f(J3) = 3 + 3 = 6,   h(J4) = 4, f(J4) = 1 + 4 = 5 (pierwszy)

Wykaz(1) = { ‚S', f_{drugi}(S)= 6 },   Aktualny = ‚J4'

Rozwiązanie 10 D): RTA* (4)

<u>Iteracja 4:</u>
TTest(‚J4') = False
FOR Następniki (‚J4')DO
„Podgląd w przód" →
h(J3) = 3, f(J3) = 1 + 3 = 4 (pierwszy)
h(S) = 6, f(S) = -1 + 6 = 5 (drugi)
Wykaz(3) = { ‚J4', f_{drugi}(J4)= 5 }
Aktualny = ‚J3'

<u>Iteracja 5:</u>
TTest(‚J3') = False
FOR Następniki (‚J3')DO
„Podgląd w przód" →
h(J8) = 0, f(J8) = 3 + 0 = 3 (pierwszy)
h(J4) = 5, f(J4) = -1 + 5 = 4 (drugi)
Wykaz(4) = { ‚J3', f_{drugi}(J3) = 4 },
Aktualny = ‚J8'



Rozwiązanie 10 D): RTA* (5)

<u>Iteracja 6:</u>
TTest(,J8') = True → Koniec [p=(J8<-J3<-J4<-S), f(p) = 5]

<u>Komentarz:</u>
Heurystyka jest dopuszczalna i spójna →
znaleziono optymalne rozwiązanie.