KECERDASAN BUATAN

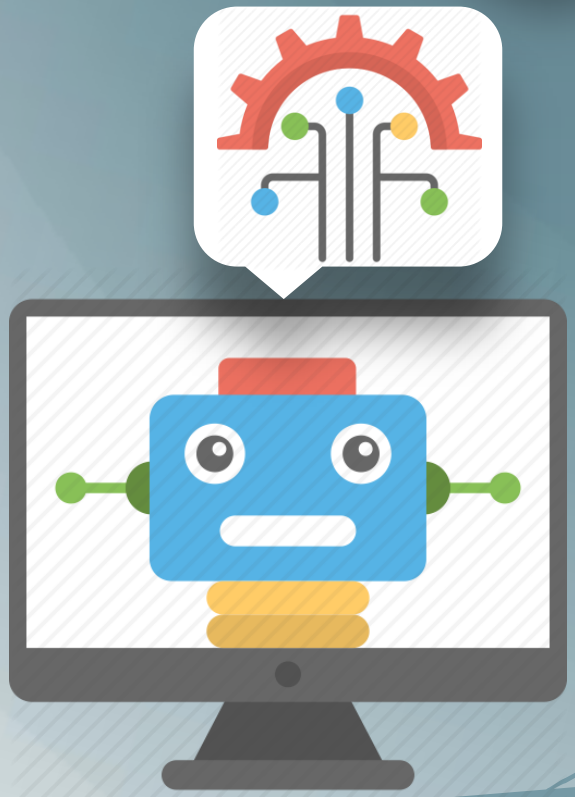
Pencarian Heuristik (Informed Search)

AMALIA NURANI B.

2019

POKOK BAHASAN

- Pencarian Heuristik
- Teknik Pencarian Heuristik



Pencarian Heuristik

WHY??

Pencarian buta tidak selalu dapat diterapkan dengan baik.

Kelemahan Blind (un-informed) Search:

- Waktu aksesnya yang cukup lama
- Besarnya memori yang dibutuhkan.

Contoh:

Breadth First Search ➔ Membangkitkan simpul berikutnya berdasarkan urutan level.

Masalahnya?

Dibutuhkan memori yang sangat besar untuk sebuah masalah yang sederhana.

Pencarian Heuristik

- Kelemahan ini sebenarnya dapat diatasi jika ada informasi tambahan (fungsi heuristik) dari domain yang bersangkutan.

Definisi Fungsi Heuristik

Heuristik

- “Aturan praktis" yang digunakan untuk membantu memandu pencarian, seringkali aturan tersebut dipelajari/ditentukan melalui pengalaman (per kasus).

- Heuristik didefinisikan sebagai aturan untuk memilih cabang-cabang dalam ruang keadaan yang paling tepat untuk mencapai solusi permasalahan yang dapat diterima

Definisi Fungsi Heuristik

Fungsi Heuristik

- Fungsi yang diterapkan pada keadaan di ruang pencarian untuk menunjukkan kemungkinan sukses jika keadaan tersebut dipilih.
- Tujuan dari sebuah fungsi heuristik adalah untuk memandu proses pencarian tujuan yang menguntungkan dengan menganjurkan jalur yang mana yang diikuti pertama kali ketika tersedia lebih dari satu tujuan.
- Setelah proses berlangsung, akan bisa dihitung sebuah fungsi heuristik yang sempurna dengan cara melakukan sebuah pencarian yang lengkap dari simpul dalam pertanyaan (ruang) dan menentukan apakah fungsi ini menuju ke sebuah solusi yang baik.

Definisi Fungsi Heuristik

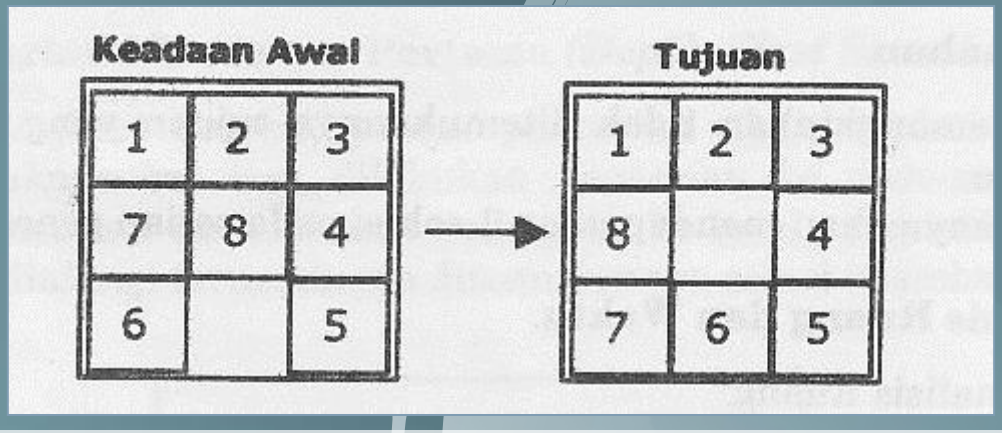
Pencarian Heuristik

- Diberi ruang pencarian, kondisi saat ini, dan status tujuan
- Menghasilkan semua successor dan mengevaluasi masing-masing dengan fungsi heuristik yang telah dipilih
- Pilih langkah yang menghasilkan nilai heuristik terbaik

Note:

- Fungsi heuristik dapat dihasilkan untuk sejumlah masalah seperti game, tetapi bagaimana dengan situasi perencanaan atau diagnostik?

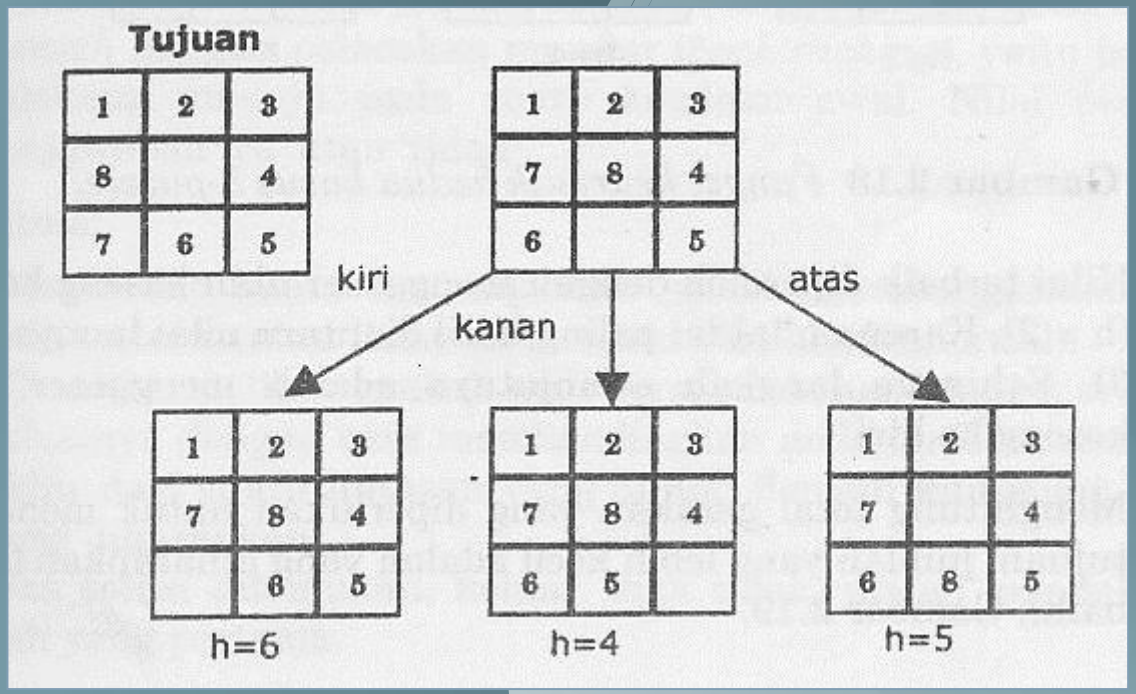
Contoh Fungsi Heuristik



Misalkan pada kasus 8-puzzle

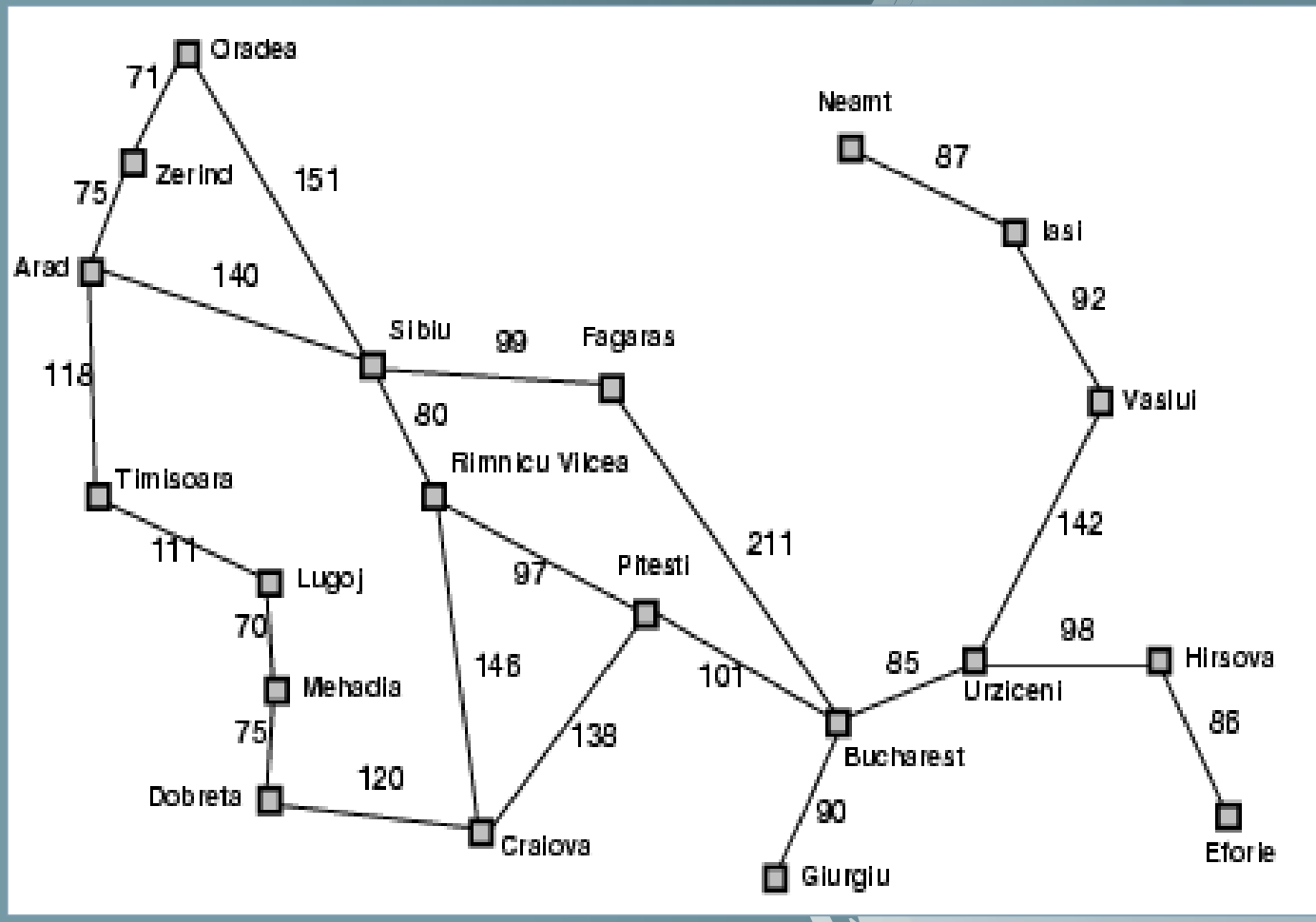
- ✓ Ada 4 operator yang dapat digunakan untuk menggerakkan dari satu keadaan (state) ke keadaan yang baru.
  - Geser ubin kosong ke kiri
  - Geser ubin kosong ke kanan
  - Geser ubin kosong ke atas
  - Geser ubin kosong ke bawah

Fungsi Heuristik



- Informasi yang diberikan dapat berupa jumlah ubin yang menempati posisi yang benar. Jumlah yang lebih banyak adalah yang diharapkan.
- Sehingga langkah selanjutnya yang harus dilakukan adalah menggeser ubin kosong ke kiri.

Fungsi Heuristik



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- Informasi yang diberikan berupa *straight-line distance* (jarak dalam garis lurus) antara tiap kota dengan Bucharest.

Teknik Pencarian Heuristik (Informed Search)

- Informed/ Heuristic Search
  - Hill Climbing
    - Simple HC
    - Steepest Ascent HC
  - Best First Search
    - A* (A Star)
    - Greedy Best First
  - Simulated Annealing
  - Iterative Deepening A*

Hill Climbing

Terdapat 2 jenis algoritma HC yaitu:

a) **Simple Hill Climbing**
b) **Steepest-Ascent Hill Climbing**

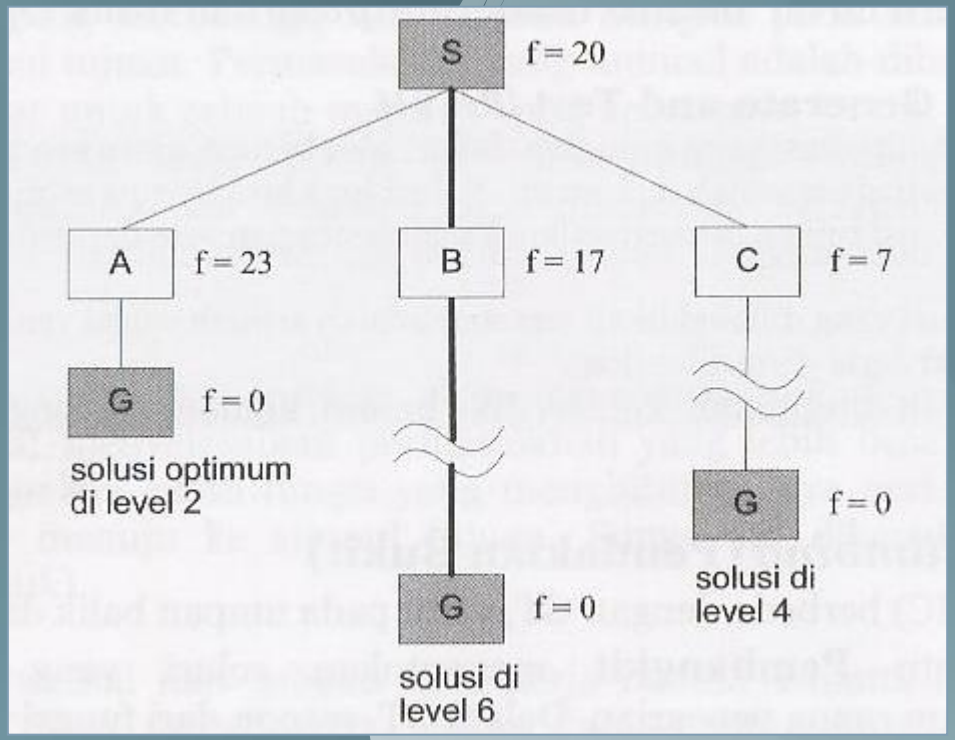
Simple Hill Climbing

Simple HC, langsung memilih *new state* yang memiliki jalur yang lebih baik daripada jalur-jalur sebelumnya tanpa memperhitungkan jalur-jalur lain yang lebih baik.

Algoritma Simple Hill Climbing

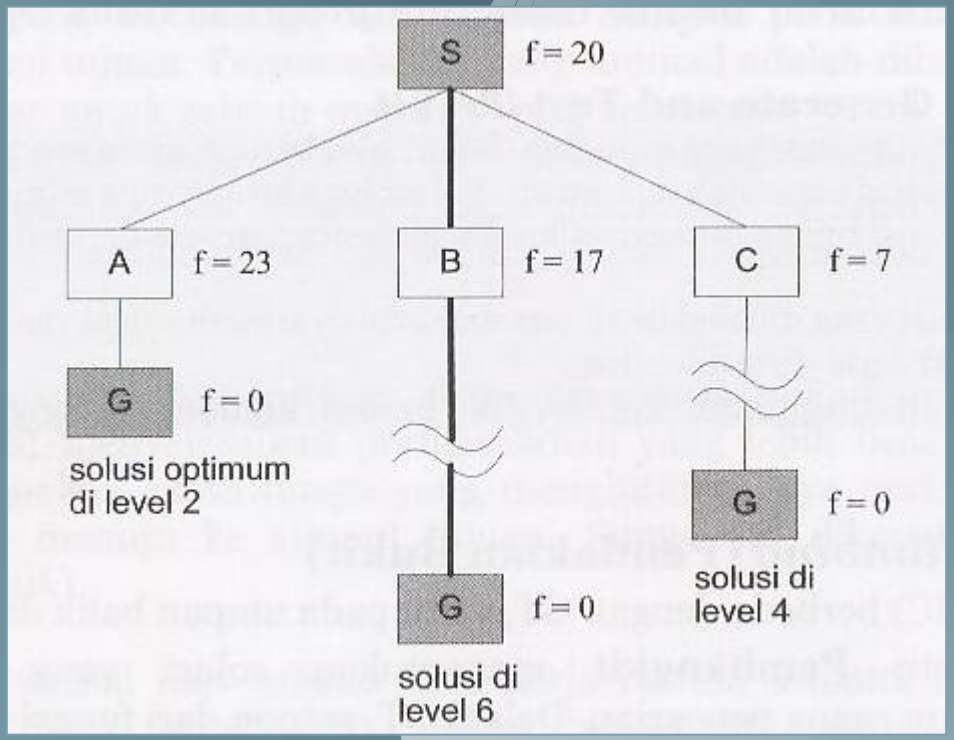
1) Evaluasi state awal, jika state awal sama dengan tujuan, maka proses berhenti. Jika tidak sama dengan tujuan maka lanjutkan proses dengan membuat state awal sebagai state sekarang.
2) Kerjakan langkah berikut sampai solusi ditemukan atau sampai tidak ada lagi operator baru yang dapat digunakan dalam state sekarang:
   a) Cari sebuah operator yang belum pernah digunakan dalam state sekarang dan gunakan operator tersebut untuk membentuk state baru.
   b) Evaluasi state baru.
      i) Jika state baru adalah tujuan, maka proses berhenti
      ii) Jika state baru tersebut bukan tujuan tetapi state baru lebih baik daripada state sekarang, maka buat state baru menjadi state sekarang.
      ii) Jika state baru tidak lebih baik daripada state sekarang, maka lanjutkan ke langkah sebelumnya.

Contoh Kasus Simple Hill Climbing



- S menyatakan intial state, sedangkan G menyatakan goal state.
- Variable f di setiap state menyatakan biaya antara state tersebut dengan goal state. Nilai f pada goal state = 0.

Contoh Kasus Simple Hill Climbing

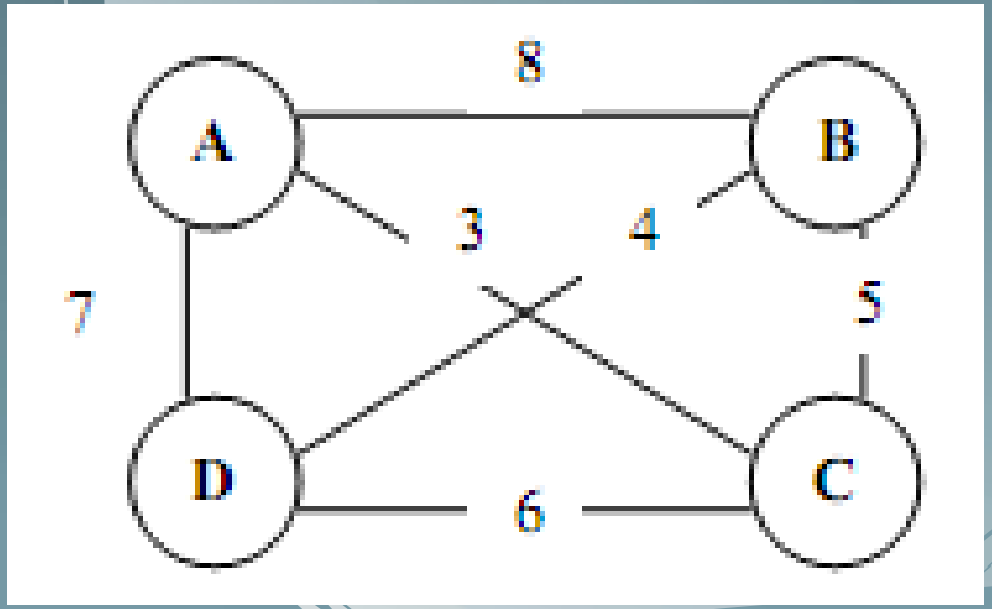


- Simple HC langsung memilih state B sebagai next state karena nilai f pada state B lebih kecil dibandingkan nilai f pada state S.
- Di sini tidak dipertimbangkan nilai f pada state C.
- Misalkan pada akhir iterasi, Simple HC mengembalikan solusi G yang berada di level 6, padahal ada solusi yang lebih baik pada level 2. Dengan demikian Simple HC tidak optimal.

Contoh Kasus Simple Hill Climbing

Contoh : Traveling Salesman Problem (TSP)

- Seorang salesman ingin mengunjungi n kota.
- Jarak antara tiap-tiap kota sudah diketahui.
- Kita ingin mengetahui rute terpendek dimana setiap kota hanya boleh dikunjungi tepat 1 kali.
- Misal ada 4 kota dengan jarak antara tiap-tiap kota seperti berikut ini :



Contoh Kasus Simple Hill Climbing

- Ruang keadaan berisi semua kemungkinan lintasan yang mungkin.
- Operator digunakan untuk menukar posisi kota-kota yang bersebelahan.
- Fungsi heuristik yang digunakan adalah panjang lintasan yang terjadi.
- Operator yang akan digunakan adalah menukar urutan posisi 2 kota dalam 1 lintasan. Bila ada n kota, dan ingin mencari kombinasi lintasan dengan menukar posisi urutan 2 kota, maka akan didapat sebanyak :

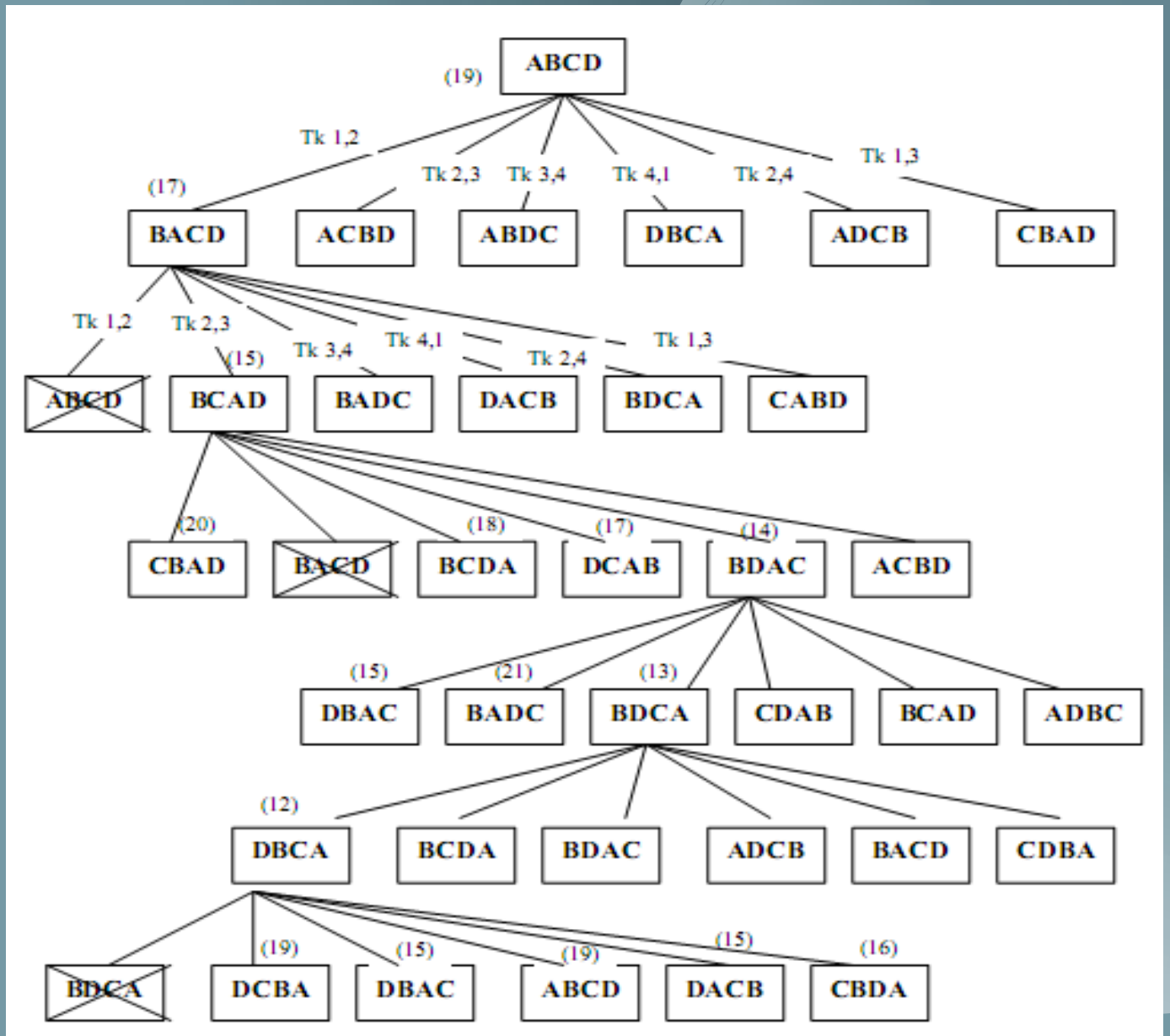
$$\frac{n!}{2!(n-2)!} = \frac{4!}{2!(4-2)!} = 6 \text{ kombinasi}$$

Contoh Kasus Simple Hill Climbing

6 kombinasi tsb digunakan sbg operator :

- Tukar 1,2 = menukar urutan posisi kota ke – 1 dengan kota ke – 2
- Tukar 2,3 = menukar urutan posisi kota ke – 2 dengan kota ke – 3
- Tukar 3,4 = menukar urutan posisi kota ke – 3 dengan kota ke – 4
- Tukar 4,1 = menukar urutan posisi kota ke – 4 dengan kota ke – 1
- Tukar 2,4 = menukar urutan posisi kota ke – 2 dengan kota ke – 4
- Tukar 1,3 = menukar urutan posisi kota ke – 1 dengan kota ke – 3

Contoh Kasus Simple Hill Climbing



Steepest Ascent HC

- Steepest-Ascent HC, akan mengevaluasi semua state yang berada di bawah current state dan memilih state dengan jalur yang hasilnya paling baik.

- Perbedaan antara Simple HC dan Steepest Ascent HC:

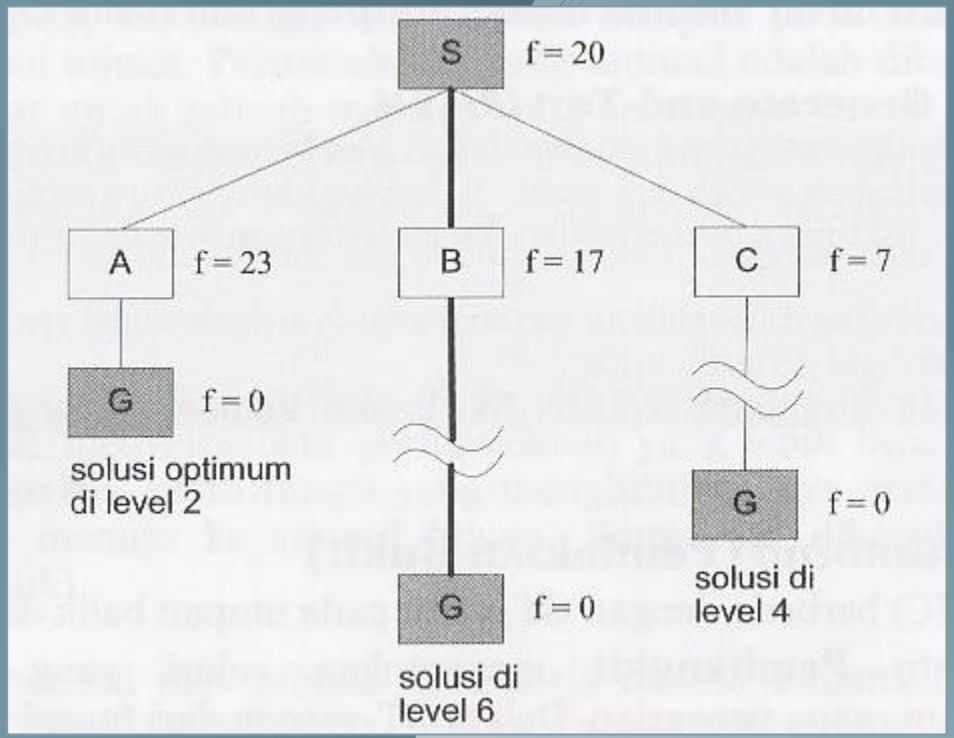
→Simple: Node pertama yang jaraknya terdekat dengan solusi yang dipilih.

→ Steepest: Semua suksesor dibandingkan, dan yang paling mendekati solusi yang dipilih.

Algoritma Steepest Ascent HC

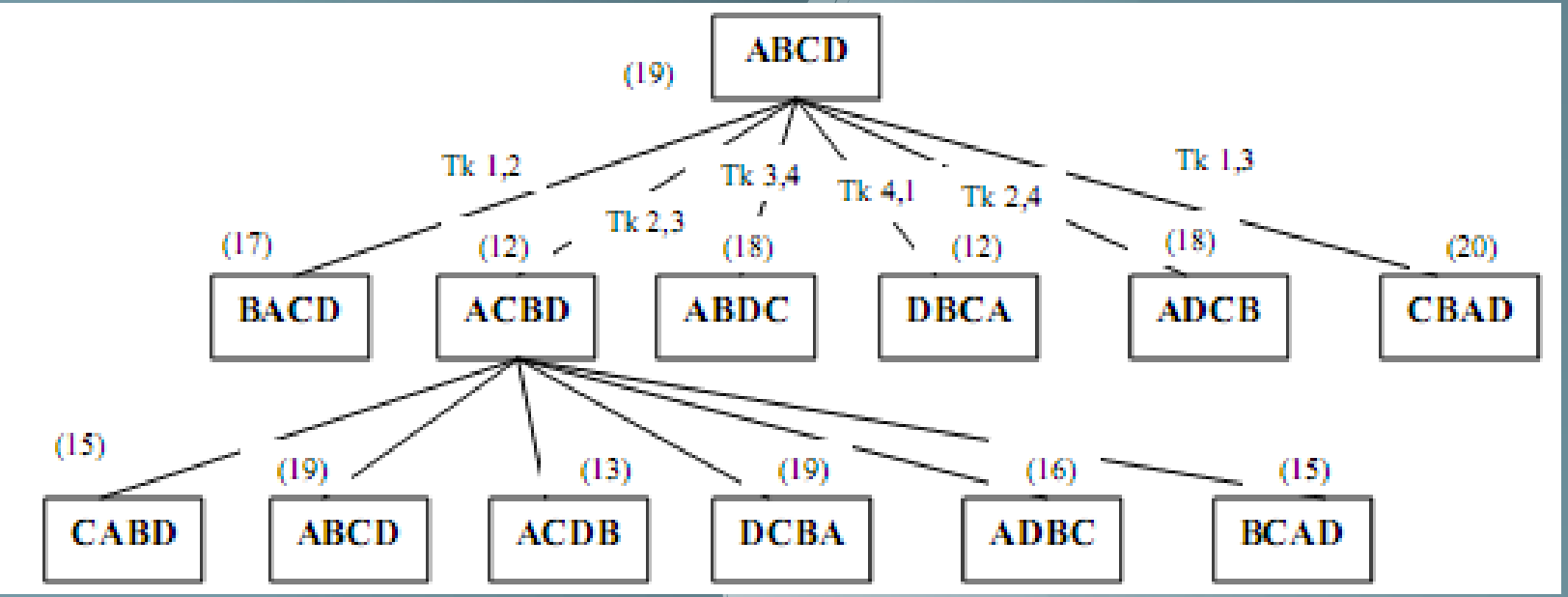
1) Evaluasi *Initial State*. Jika *Initial state* sama = *Goal* maka kembali pada *initial state* dan berhenti berproses. Jika tidak maka *initial state* tersebut jadikan sebagai *current state*.
2) Mulai dengan *current state* = *initial state*.
3) Dapatkan semua pewaris (*successor*) yang dapat dijadikan *next state pada current state*nya dan evaluasi *successor* tersebut dengan fungsi evaluasi dan beri nilai pada setiap *successor* tersebut.
4) Jika salah satu dari *successor* tersebut mempunyai nilai yang lebih baik dari *current state* maka jadikan *successor* dengan nilai yang paling baik tersebut sebagai *new current state*.
5) Lakukan operasi ini terus menerus hingga tercapai *current state* = *goal state* atau tidak ada perubahan pada *current state*nya.

Contoh Kasus Steepest Ascent HC



- Dari state S, Steepest-Ascent HC akan mengevaluasi semua state yang menjadi next state atau suksesornya, yaitu A, B, dan C. Dari ketiga suksesor tersebut dipilih suksesor dengan nilai f yang terkecil.
- State C akan dipilih sebagai suksesor S.
- Misalkan, hasil penelusuran menemukan solusi G di level 4, padahal ada solusi optimal di level 2, dalam hal ini Steepest-Ascent HC dikatakan terjebak pada solusi lokal atau local minimum. Jadi Steepest-Ascent HC juga tidak optimal.

Contoh Kasus Steepest Ascent HC



- Keadaan awal, lintasan ABCD (=19).
- Level pertama, hill climbing memilih nilai heuristik terbaik yaitu ACBD (=12) sehingga ACBD menjadi pilihan selanjutnya.
- Level kedua, hill climbing memilih nilai heuristik terbaik, karena nilai heuristik lebih besar dibanding ACBD, maka hasil yang diperoleh lintasannya tetap ACBD (=12)

Best First Search

- Best First Search menggabungkan kelebihan Depth First Search dan Breadth First Search.
- Tujuan penggabungan:
  →Menelusuri 1 jalur pada satu saat, tetapi dapat berpindah ketika jalur lain terlihat lebih menjanjikan dari jalur yang sedang ditelusuri.

  →Jalur yang menjanjikan diperoleh dengan memberikan skala prioritas pada setiap state yang dihasilkan dengan fungsi heuristik.
- Bedanya dengan Steepest Ascent HC:
  →Steepest memilih simpul yang memiliki biaya terkecil diatara suksesor saudaranya (sibling).

  →Best First Search memilih simpul baru yang memiliki biaya terkecil diantara semua leaf node (simpul pada level terdalam) yang pernah dibangkitkan.

Best First Search

- Best First Search membolehkan mengunjungi node yang lebih rendah jika ternyata node di level lebih tinggi nilai heuristiknya lebih buruk.

- Penentuan node terbaik dapat dilakukan menggunakan informasi :
  a) Biaya perkiraan → Fungsi heuristik (Current state ke Goal state)
  b) Biaya sebenarnya → Biaya Initial state → Current state)

Best First Search

Untuk menggunakan Best First Search diperlukan 2 daftar (list) dari simpul, yaitu:

a) OPEN:
Berisi simpul yang dihasilkan dari fungsi heuristik tetapi belum dievaluasi (belum dipilih), memiliki antrian prioritas dimana elemen dengan prioritas tertinggi adalah yang memiliki nilai paling baik yang dihasilkan fungsi heuristik.

b) CLOSED
Berisi simpul yang sudah dievaluasi (dipilih). List ini perlu tetap disimpan sebagai acuan saat memeriksa node baru, apakah node tersebut sudah dievaluasi atau belum.

Algoritma Best First Search

1. OPEN berisi initial state dan CLOSED masih kosong.
2. Ulangi sampai goal ditemukan atau sampai tidak ada lagi node di dalam OPEN:
   a) Ambil simpul terbaik yang ada di OPEN.
   b) Jika simpul tersebut sama dengan goal, maka sukses.
   c) Jika tidak, masukkan simpul tersebut ke dalam CLOSED.
   d) Bangkitkan semua suksesor dari simpul tersebut.
   e) Untuk setiap suksesor kerjakan:
      I. Jika suksesor tersebut belum pernah dibangkitkan, evaluasi suksesor tersebut, tambahkan ke OPEN, dan catat parent atau orang tuanya.
      II. Jika suksesor tersebut sudah pernah dibangkitkan, ubah parent-nya jika jalur melalui parent ini lebih baik daripada jalur melalui parent yang sebelumnya. Selanjutnya perbarui biaya untuk suksesor tersebut dan nodes lain yang berada di level bawahnya.

Best First Search

Terdapat dua jenis Best First Search:

a) **Greedy Best First Search:**
   Hanya memperhitungkan *estimated cost* (biaya perkiraan) saja.

b) **A* (A Star)**
   Memperhitungkan gabungan 2 biaya, biaya sebenarnya dan biaya perkiraan.

Greedy Best First Search

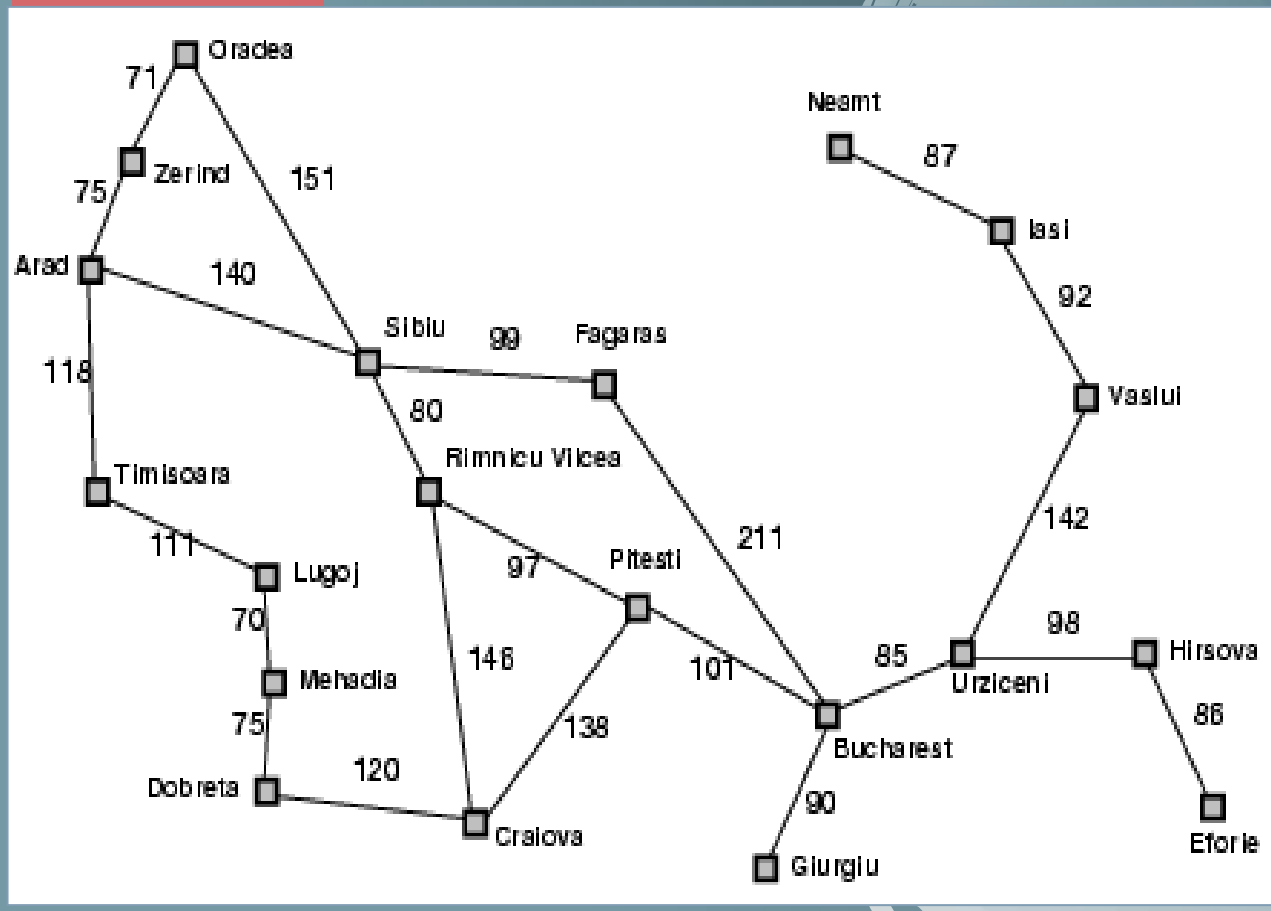
- Greedy Best First Search hanya memperhitungkan biaya perkiraan (*estimated cost*) saja, yakni:

> ***f(n) = h(n)***
>
> **di mana *h(n)*= perkiraan biaya dari simpul n ke goal.**

- Biaya yang sebenarnya (*actual cost*) tidak diperhitungkan.

Contoh Kasus Greedy Best First Search

Diketahui

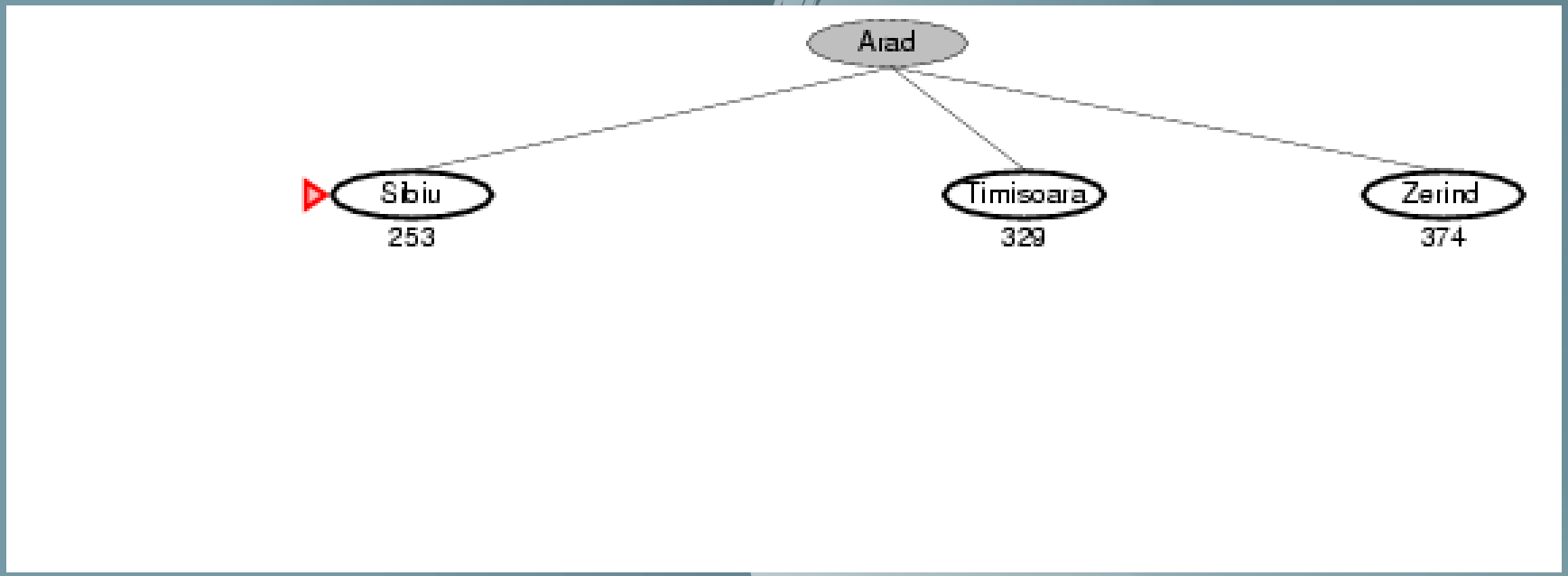


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

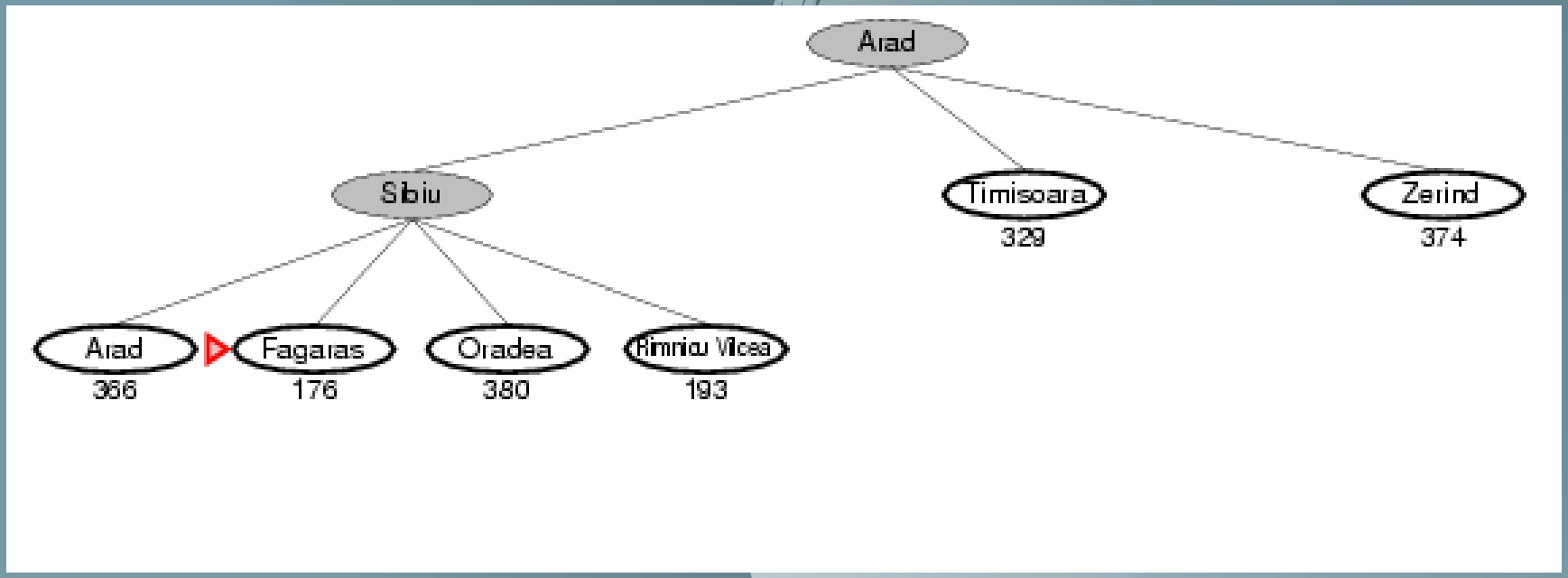
f(n) = h(n)

di mana *h(n)*= perkiraan biaya dari simpul n ke goal (estimated cost).

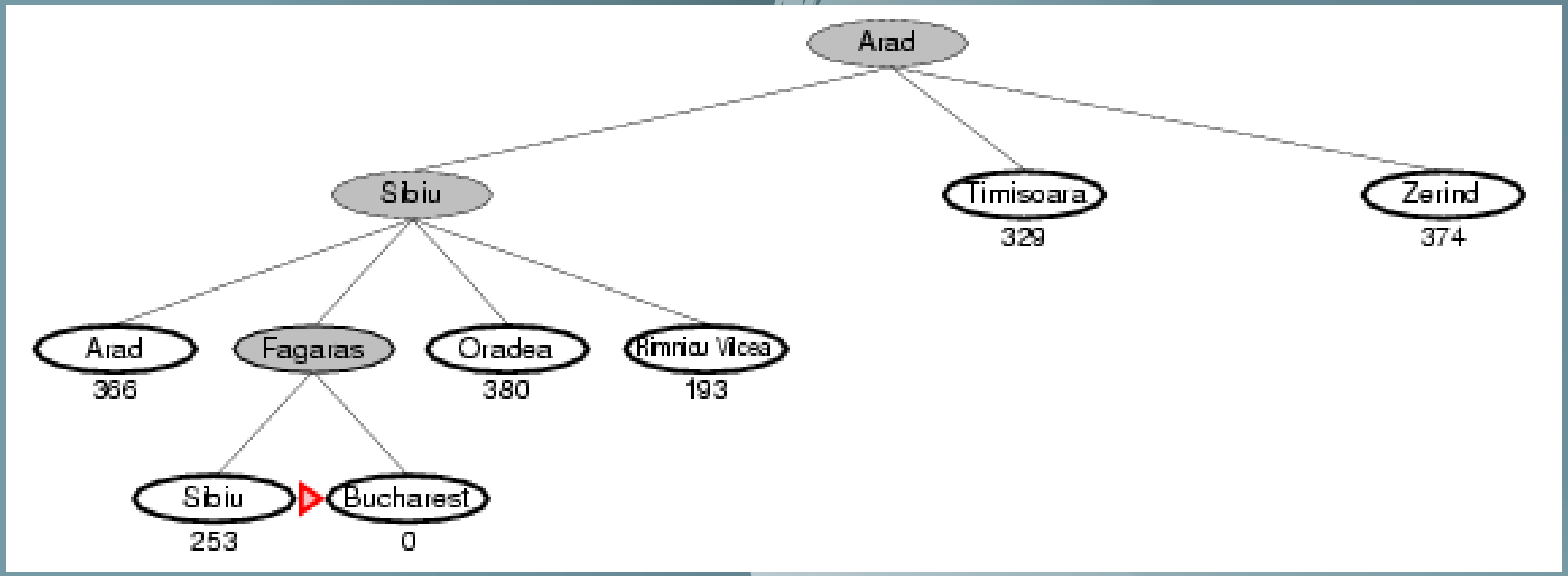
Contoh Kasus Greedy Best First Search



Contoh Kasus Greedy Best First Search



Contoh Kasus Greedy Best First Search



- Solusi yang ditemukan adalah: Arad, Sibiu, Fagaras, Bucharest.
- Ternyata solusi ini tidak optimal karena ada solusi lain yang lebih baik, yaitu: Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest yang lebih pendek 32 kilometer.

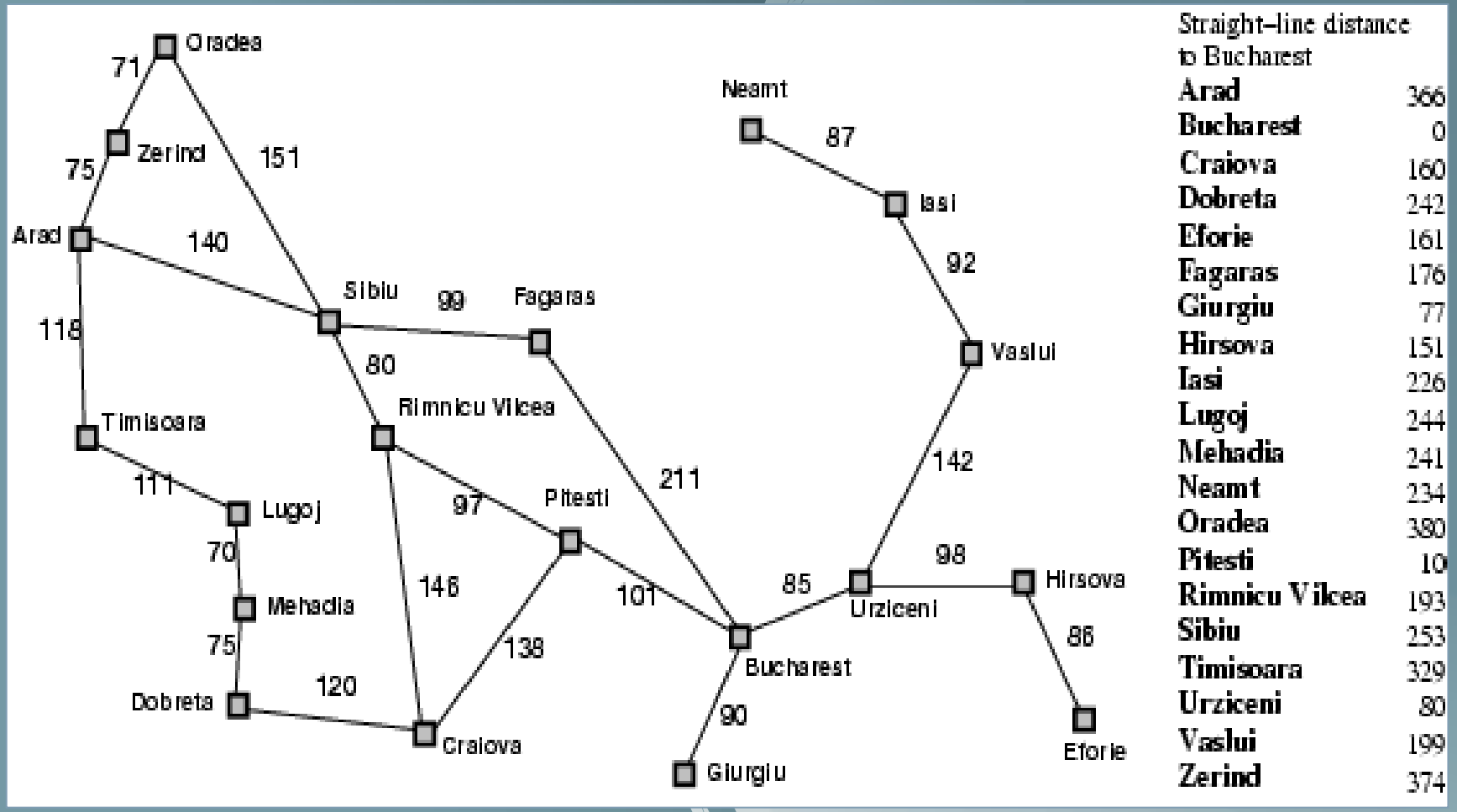
A* Search

- Algoritma ini memperhitungkan biaya dari biaya sebenarnya ditambah dengan biaya perkiraan.
- Dalam notasi matematika dituliskan sebagai:

  $$f(n) = g(n) + h(n)$$

  - $g(n)$ = biaya sebenarnya untuk mencapai simpul n
  - $h(n)$ = perkiraan biaya dari simpul n ke goal.
  - $f(n)$ = perkiraan total biaya jalur yang melalui simpul n ke goal.
- Dengan perhitungan biaya seperti ini, algoritma A* adalah complete dan optimal.

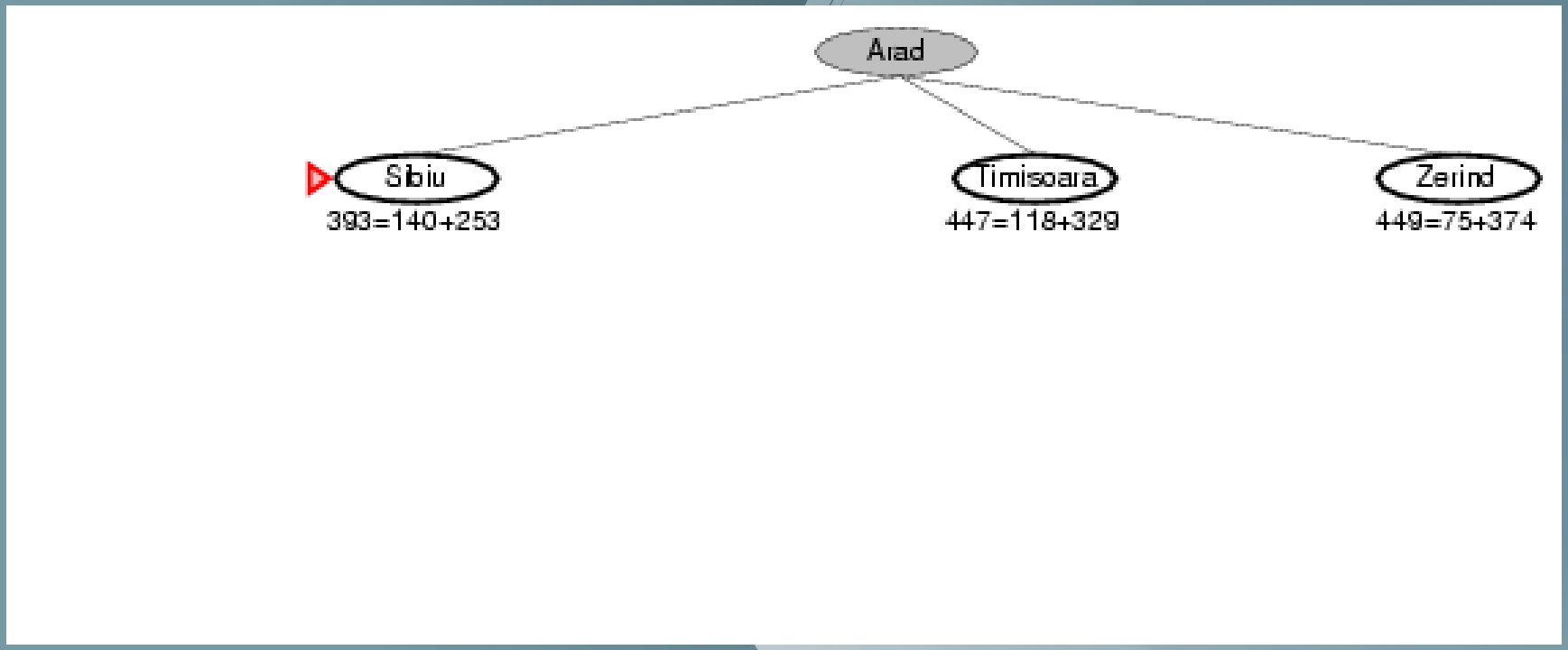
Contoh A* Search



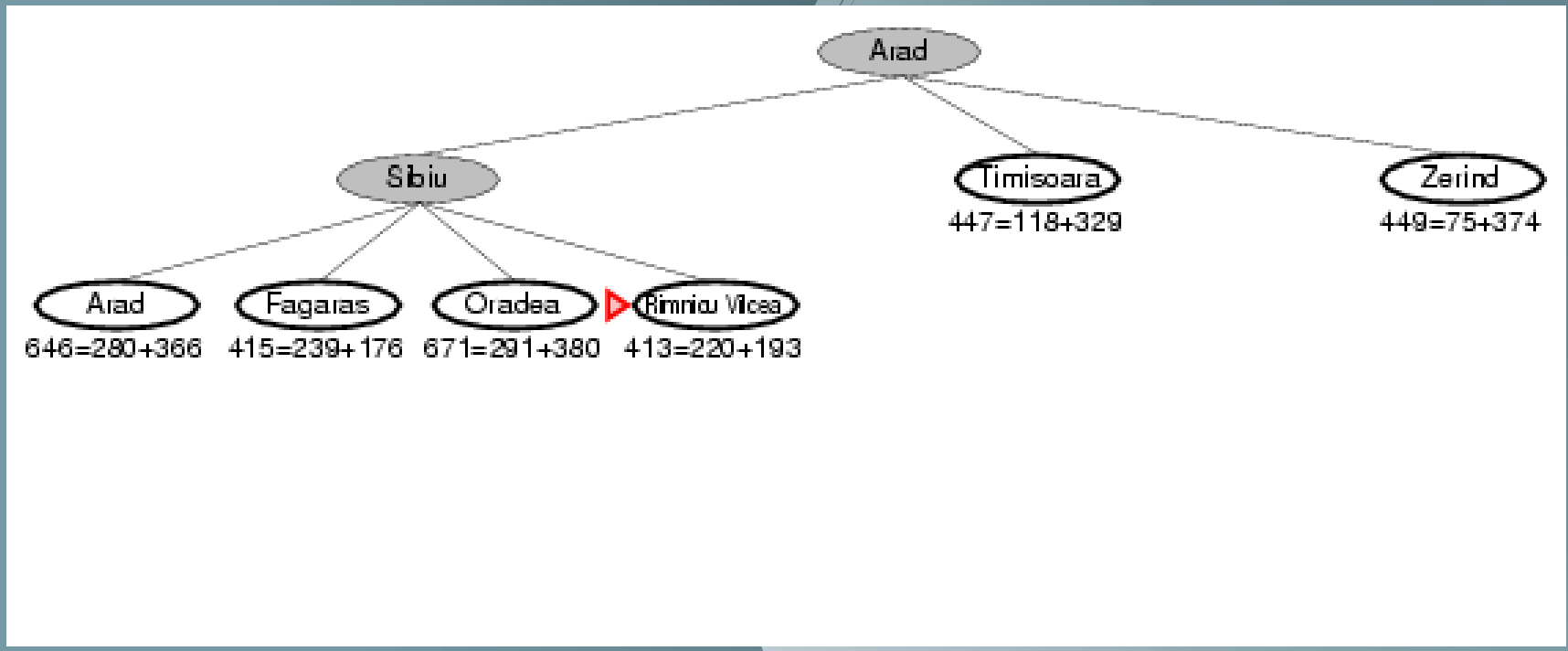
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- f(n) = g(n) + h(n)
- g(n) = biaya sebenarnya untuk mencapai sebuah node (kota) n
- h(n) = jarak garis lurus dari node n ke goal (Bucharest)

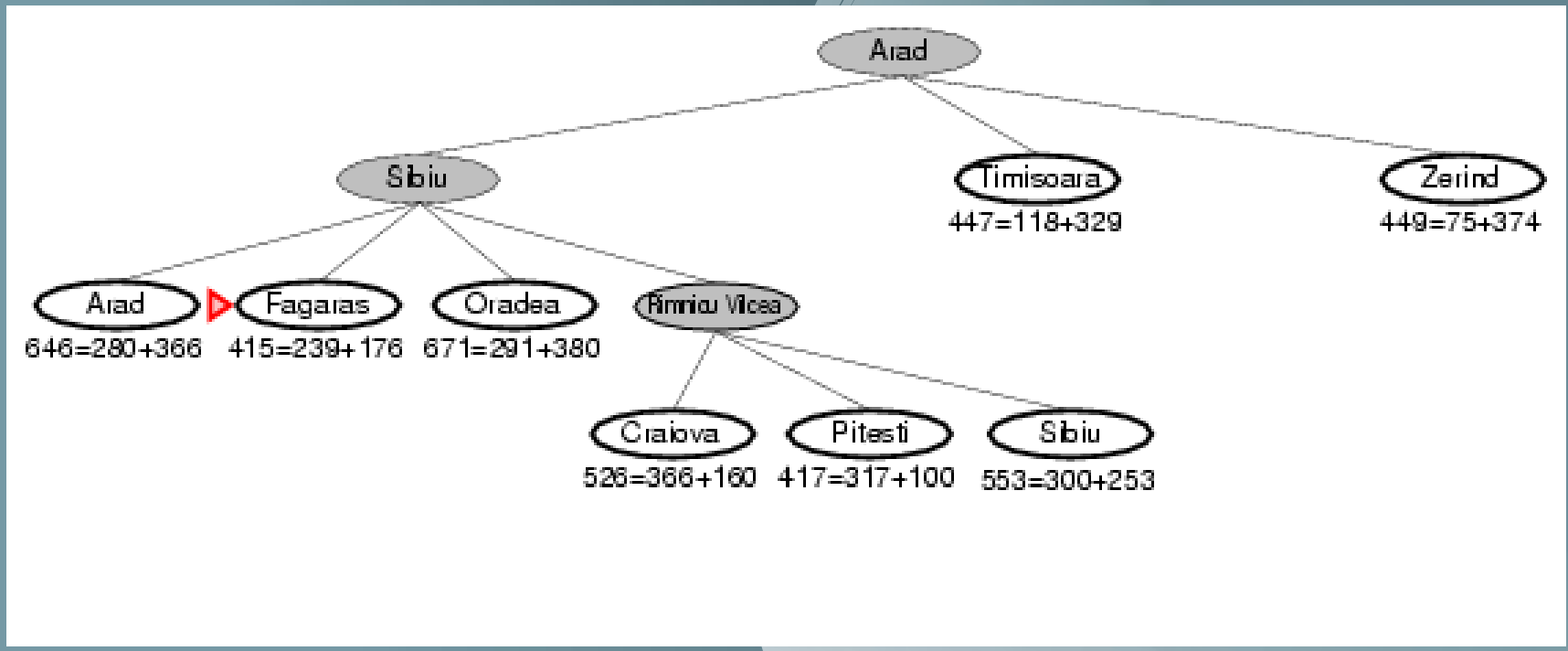
Contoh A* Search



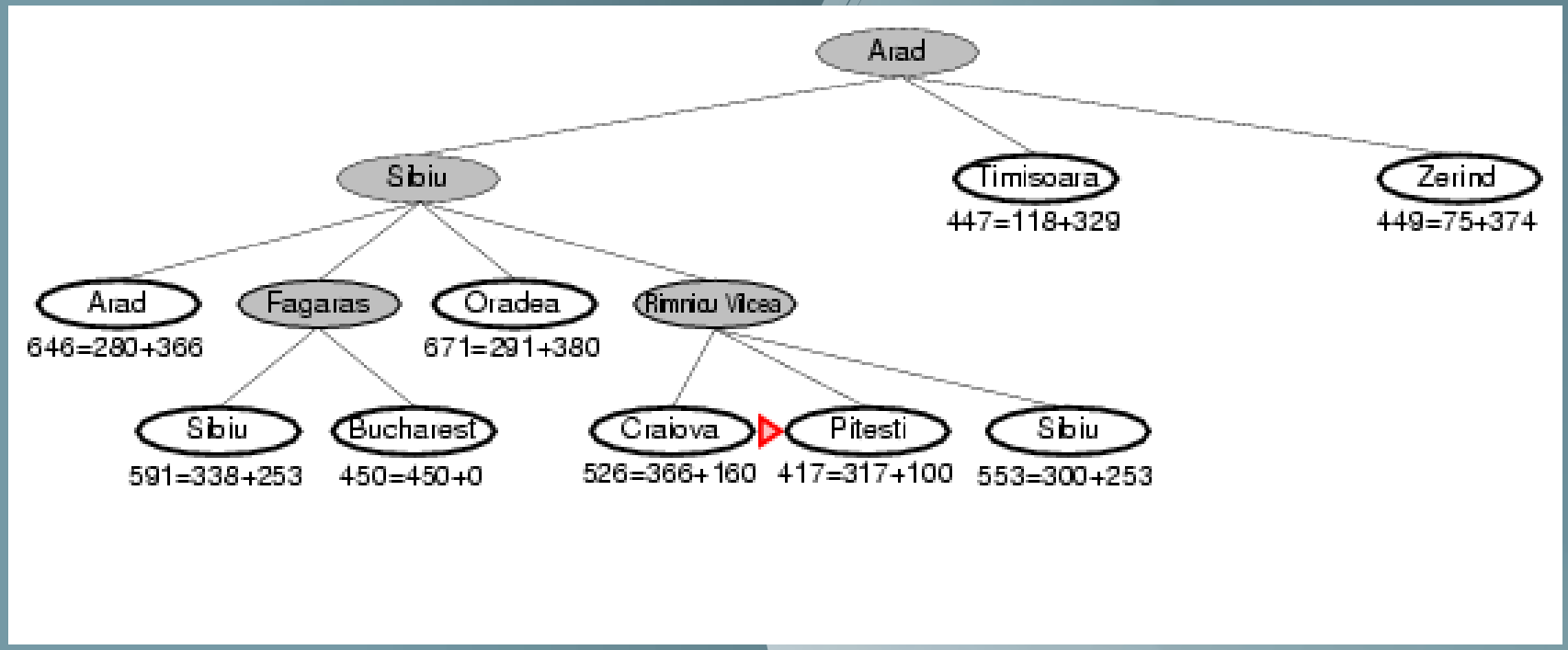
Contoh A* Search



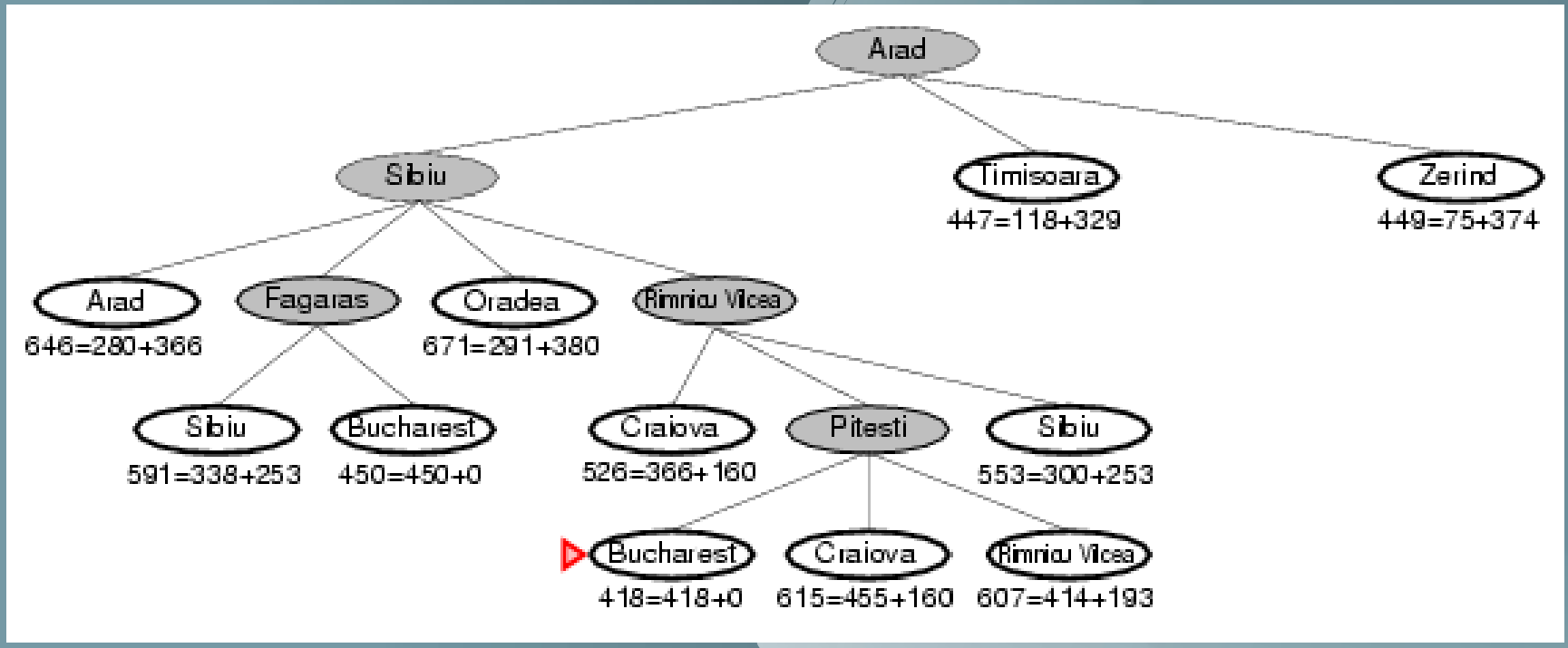
Contoh A* Search



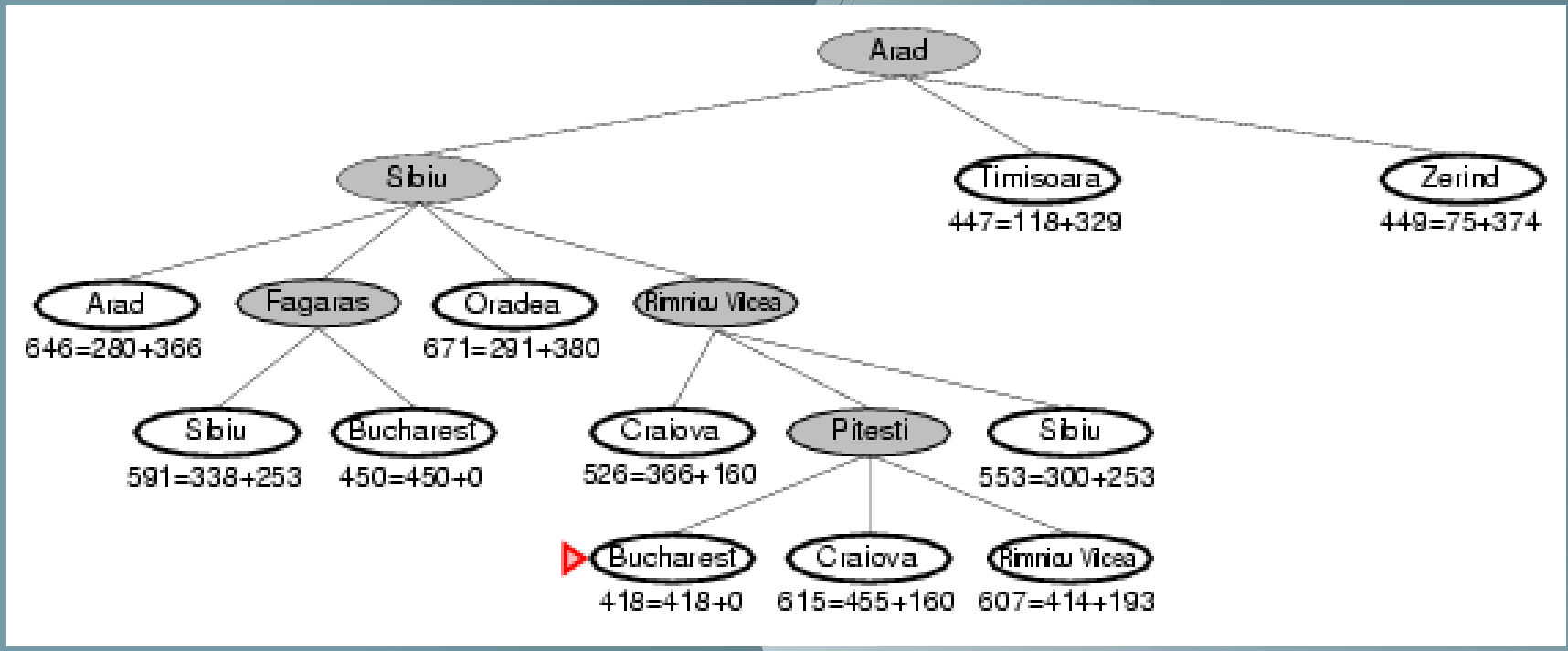
Contoh A* Search



Contoh A* Search



Contoh A* Search



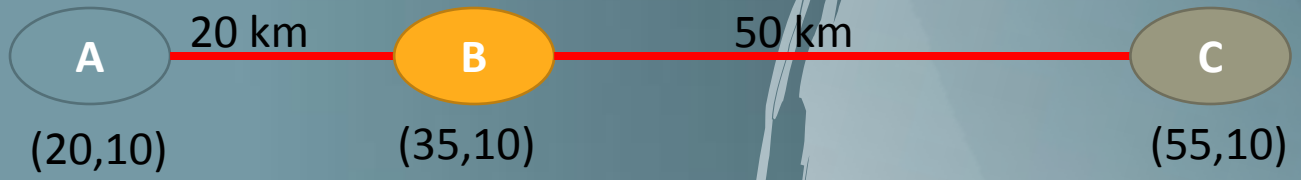
- Solusi yang ditemukan adalah solusi optimal, yaitu: Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest.

Pemilihan Fungsi Heuristik

- Fungsi heuristik sangat menentukan hasil dari teknik-teknik heuristic search.
- Fungsi heuristik dapat diterima (admissible) jika biaya perkiraan yang dihasilkan tidak melebihi biaya sebenarnya.
- Ketika fungsi heuristik memberikan biaya perkiraan yang melebihi biaya sebenarnya (overestimate), maka proses pencarian bisa tersesat dan membuat heuristic search menjadi kurang optimal.

Contoh Kasus Penggunaan Fungsi Heuristik

Masalah: Pencarian Rute Terpendek



- Biaya sebenarnya = panjang jalan raya. Fungsi heuristik yang dapat digunakan adalah jarak garis lurus.

$$d_{ab} = \sqrt{(y_b - y_a)^2 + (x_b - x_a)^2}$$

Diperoleh: d_{ab}**= 15 ,** d_{bc} **= 20**

- d_{ab} mendekatijaraksebenarnyadan d_{bc} meskiagakjauhdarija raksebenarnyatapinilainyalebihkecil.

Contoh Kasus Penggunaan Fungsi Heuristik

Masalah: 8-Puzzle

1	2	3
8	7	4
6		5

Initial state

1	2	3
4	5	6
7	8	

Goal state

Terdapat 2 jenis fungsi heuristik yang bisa digunakan:

- h_1 : jumlah kotak yang posisinya salah. ➔ Angka 1,2,3 sudah berada di posisi yang benar. Sedangkan lima angka yang lain berada di posisi yang salah. Jadi h_1 = 5
- h_2 : jumlah langkah yang diperlukan masing-masing kotak menuju posisi yang benar di *goal* state (*City block distance* atau *Manhattan distance*). ➔ 1, 2 , dan 3 membutuhkan 0 langkah. 4,5,7, dann 8 = 2 langkah. Sedangkan 6 = 3 langkah.
  Sehinggah_2 = 0+0+0+2+2+3+2+2=11

TUGAS 3

DEADLINE PENGUMPULAN:
KAMIS, 28 MARET 2019 PUKUL 23.00.

1) Jelaskan dengan rinci, algoritma serta contoh kasus dari teknik pencarian di bawah ini:

a. **Simulated Annealing**
b. **Iterative Deepening A* (IDA*)**

2) Cari jurnal penelitian mengenai:

a) **Hill Climbing**
b) **Best First Search**

Kirim ke: **Google Classroom** dengan judul AI3_Stambuk_Nama. Ex: AI3_E1E108_Amalia

Contoso

Terima kasih