Uninformed Search

Ali Akbar Septiandri

November 28, 2017

Universitas Al Azhar Indonesia

Daftar isi

1. Masalah Carian

2. Pohon Carian

3. Program Dinamis

Masalah Carian

A **farmer** wants to get his **cabbage**, **goat**, and **wolf** across a river. He has a boat that only holds two. He cannot leave the cabbage and goat alone or the goat and wolf alone. How many river crossings does he need?

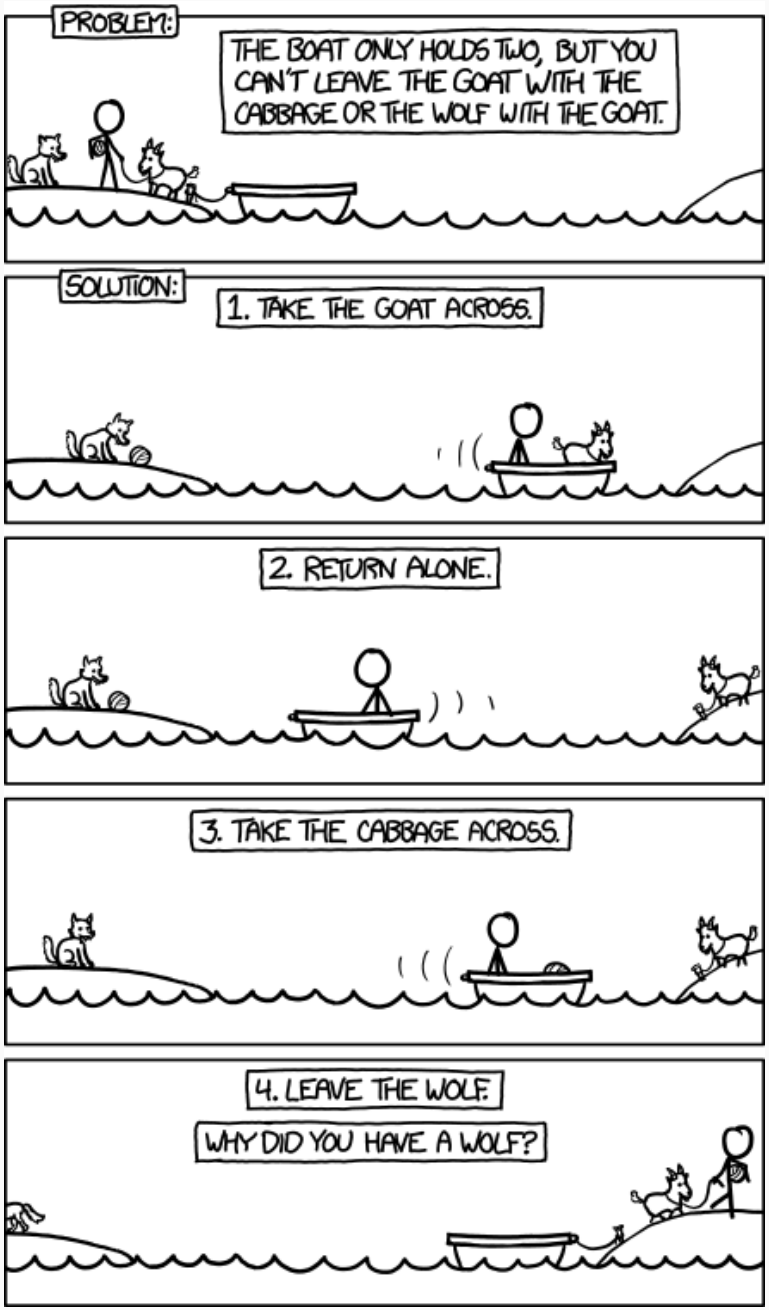


Figure 1: Buat apa membawa serigala? (Komik dari XKCD)

Aplikasi: Pencarian rute

Figure 2: Apa saja komponen PEAS-nya?

Aplikasi: Gerakan robot

Figure 3: Apa saja komponen PEAS-nya?

Aplikasi: Menyelesaikan puzzle

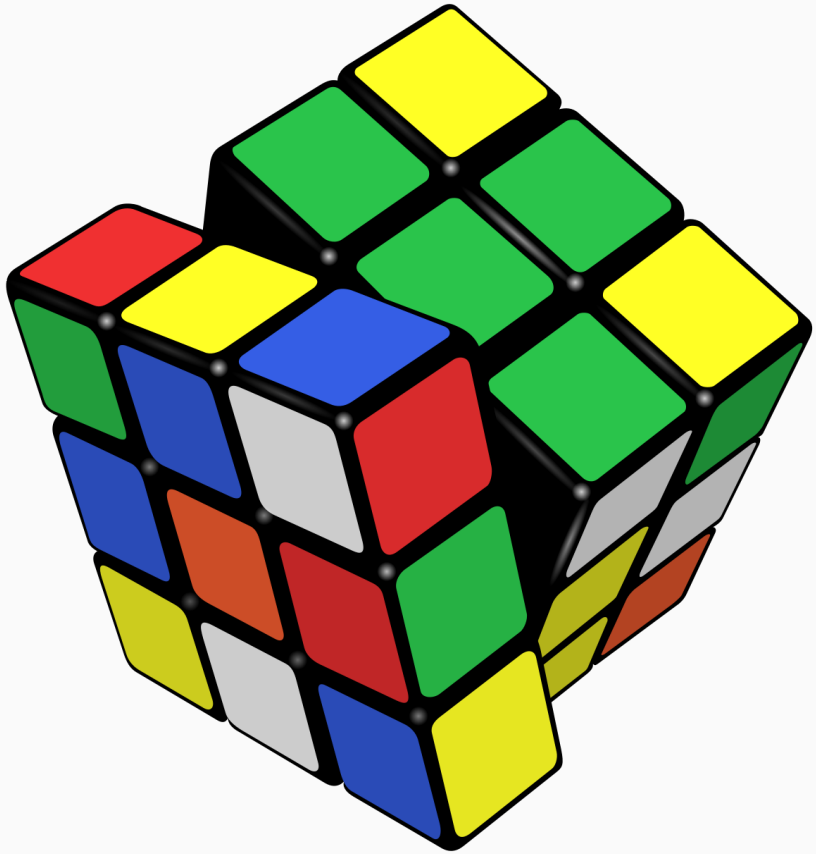


Figure 4: Mengapa menjadi kasus carian?

Melebihi refleks

Klasifikasi

(model berbasis refleks)

$$x \rightarrow f \rightarrow \text{aksi tunggal } y \in \{-1, +1\}$$

Carian

(model berbasis status)

$$x \rightarrow f \rightarrow \text{urutan aksi } (a_1, a_2, a_3, ...)$$

Mengapa tidak menjadikan carian sebagai **kumpulan refleks**?

Pohon Carian

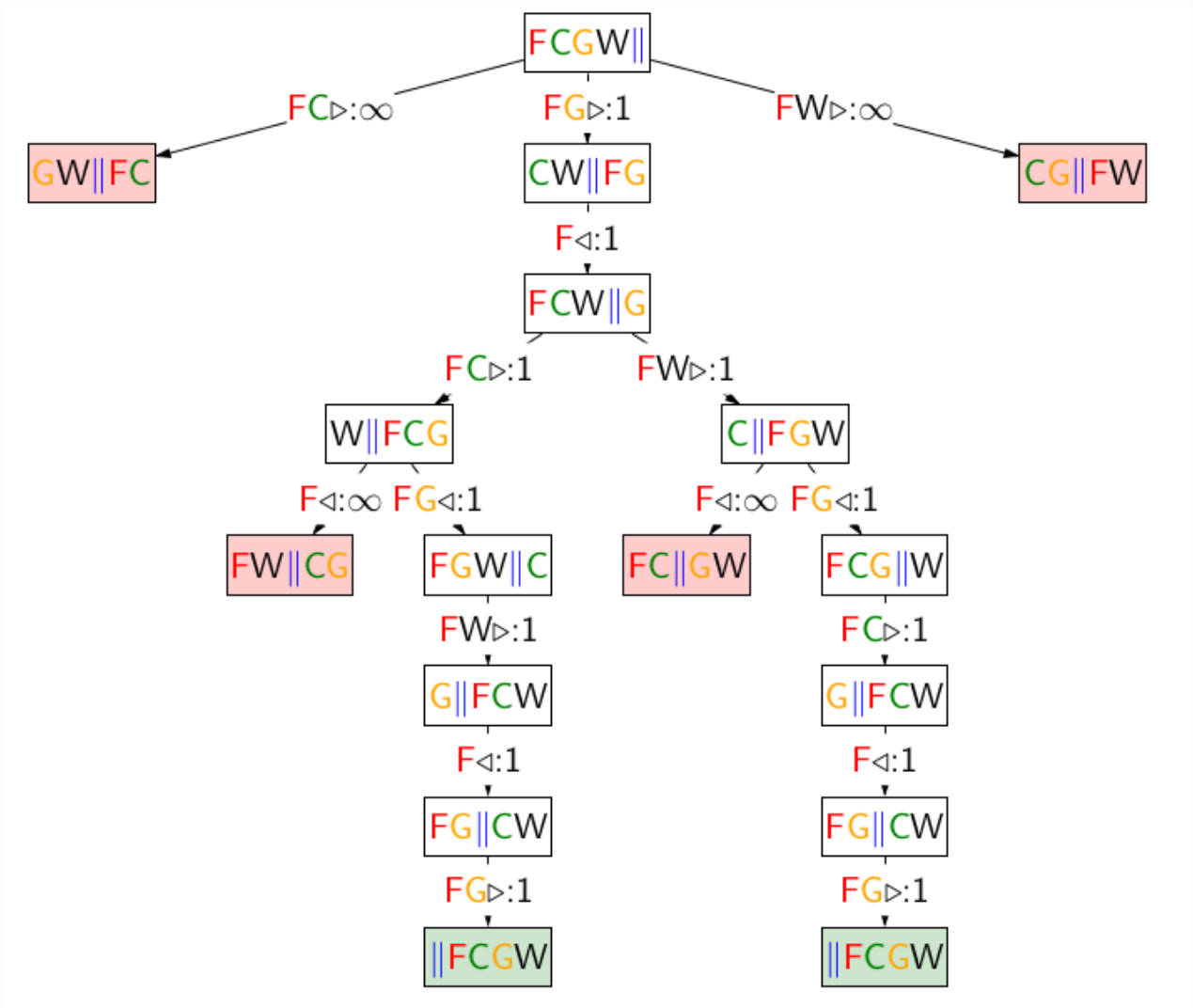
Kembali ke permulaan...

Farmer, Cabbage, Goat, Wolf

Aksi

- F$\triangleright$, F$\triangleleft$
- FC$\triangleright$, FC$\triangleleft$
- FG$\triangleright$, FG$\triangleleft$
- FW$\triangleright$, FW$\triangleleft$

Solusi


FCGW||
FC▷:∞
FG▷:1
FW▷:∞
GW||FC
CW||FG
CG||FW
F◁:1
FCW||G
FC▷:1
FW▷:1
W||FCG
C||FGW
F◁:∞
FG◁:1
F◁:∞
FG◁:1
FW||CG
FGW||C
FC||GW
FCG||W
FW▷:1
FC▷:1
G||FCW
G||FCW
F◁:1
F◁:1
FG||CW
FG||CW
FG▷:1
FG▷:1
||FCGW
||FCGW

Bagaimana cara menghasilkan pohon seperti ini?

Definisi

Kasus carian:

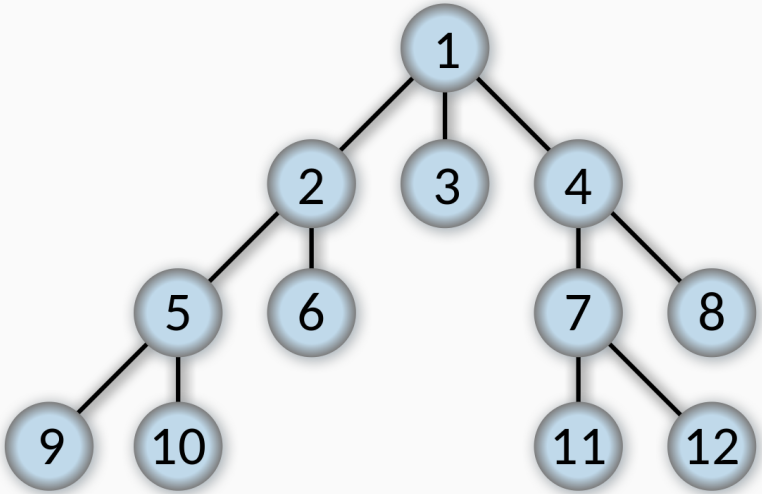
- s_{start}: kondisi awal
- Actions(s): kemungkinan aksi
- Cost(s, a): ongkos aksi
- Succ(s, a): suksesor
- IsEnd(s): kondisi akhir?

Algoritma untuk pohon carian

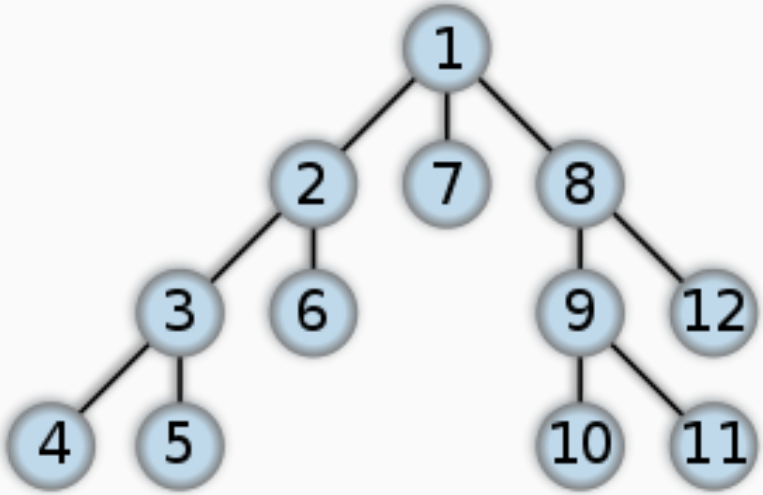
- Breadth-first search (BFS; *queue*)
- Depth-first search (DFS; *stack*)

BFS vs DFS

BFS



DFS



Iterative deepening

- Iteratively use depth-limited search with increasing depth

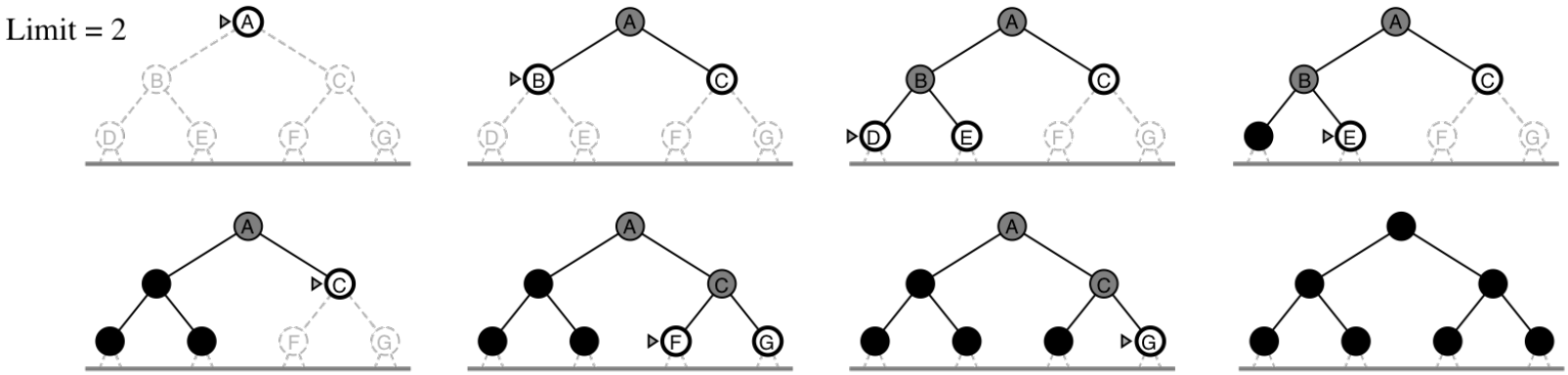
Limit = 0

Iterative deepening

- Iteratively use depth-limited search with increasing depth

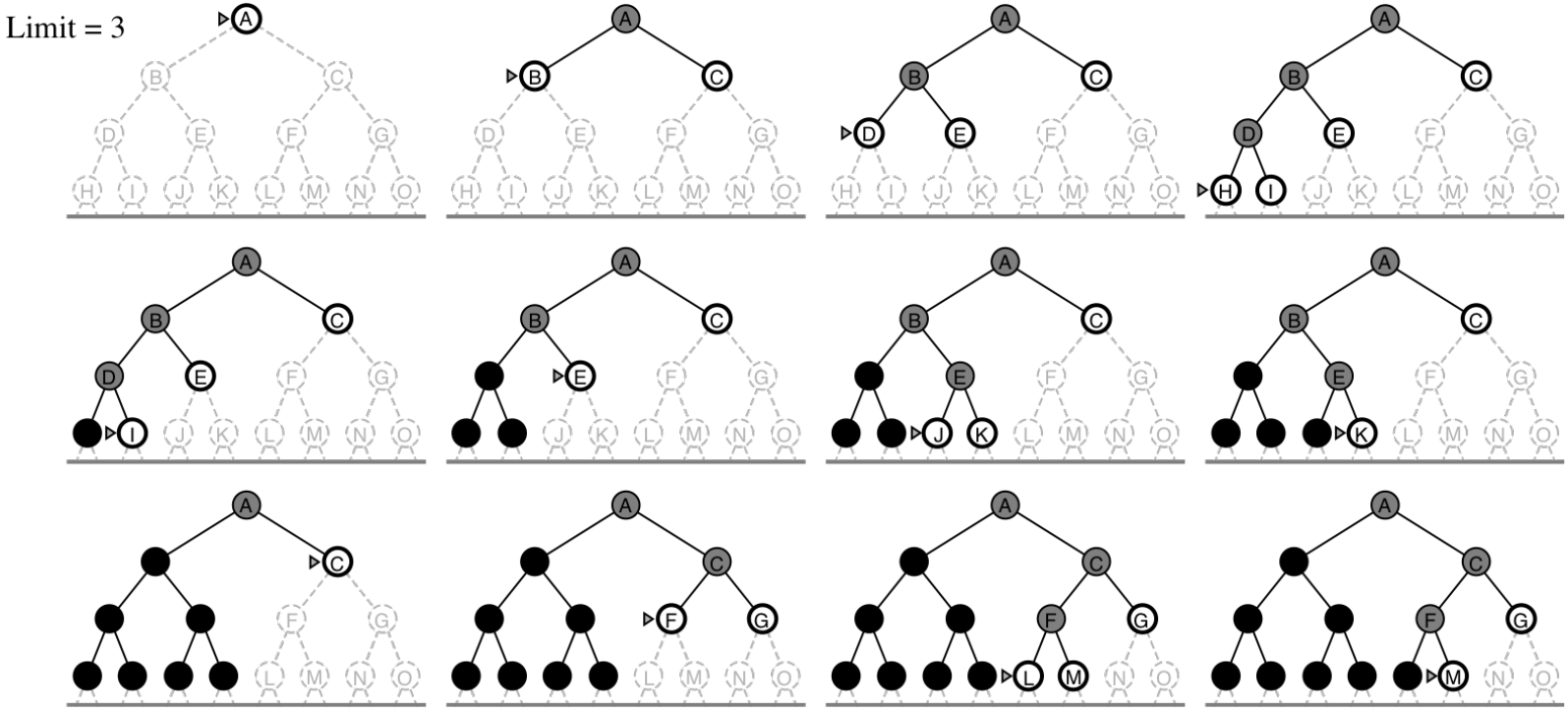
Iterative deepening

- Iteratively use depth-limited search with increasing depth



Iterative deepening

- Iteratively use depth-limited search with increasing depth



Yang perlu diperhatikan adalah **kompleksitas** dalam **ruang** dan **waktu**. Memori komputer terbatas, tapi waktu yang "berkembang" dapat diatasi dengan membiarkan program berjalan lebih lama.

Informed vs uninformed search

- *Uninformed search*: Hanya bisa membedakan kondisi tujuan (*goals*) dan bukan tujuan (*non goals*)
- *Informed search*: Mengetahui progres menuju solusi optimal

Program Dinamis

Anda membayar ke sejumlah uang ke kasir. Kasir tersebut kemudian harus memberikan kembalian berupa koin. Berapa denominasi yang harus digunakan agar koin yang diberikan minimum?

Anda dapat menggunakan solusi greedy, i.e. menggunakan pecahan terbesar sebisa mungkin.

Example (Counter example)

Diberikan denominasi 1, 5, 10, 20, 25, dan 50 sen, berapa jumlah koin minimum yang diperlukan untuk kembalian 40 sen?

Example

Diberikan denominasi 6, 5, dan 1 sen, berapa jumlah koin minimum yang diperlukan untuk kembalian 9 sen?

$$MinNumCoins(9) = \min \begin{cases} MinNumCoins(9-6)+1 = MinNumCoins(3)+1 \\ MinNumCoins(9-5)+1 = MinNumCoins(4)+1 \\ MinNumCoins(9-1)+1 = MinNumCoins(8)+1 \end{cases}$$

$$MinNumCoins(3) = \ ?$$
$$MinNumCoins(4) = \ ?$$
$$MinNumCoins(8) = \ ?$$

Solusi umum

$$MinNumCoins(money) = \min \begin{cases} MinNumCoins(money - coin_1) + 1 \\ \dots \\ MinNumCoins(money - coin_d) + 1 \end{cases}$$

Algoritma

```
RecursiveChange(money, coins)
begin
    if money = 0 then
        return 0
    end
    MinNumCoins ← ∞
    for i ← 1 to |coins| do
        if money ≥ coin_i then
            numCoins ← RecursiveChange(money − coin_i, coins)
            if numCoins + 1 < MinNumCoins then
                MinNumCoins ← numCoins + 1
            end
        end
    end
    return MinNumCoins
end
```

Algoritma tersebut dijamin benar, tetapi bisa menjadi sangat mahal dalam komputasi.

Example

Diberikan denominasi 6, 5, dan 1 sen, berapa jumlah koin minimum yang diperlukan untuk kembalian 76 sen?

Example

Diberikan denominasi 6, 5, dan 1 sen, berapa jumlah koin minimum yang diperlukan untuk kembalian 76 sen?

Solusi

Untuk kombinasi 69 sen, kita akan menghitung 6 kali.

Untuk kombinasi 30 sen, kita akan menghitung triliunan kali!

Gunakan **memoization**, i.e. ***lookup table***.

Memoization

Example

Diberikan denominasi 6, 5, dan 1 sen, berapa jumlah koin minimum yang diperlukan untuk kembalian 9 sen?

Solusi

money	0	1	2	3	4	5	...
MinNumCoins	0	1	2	3	4	1	...

Algoritma program dinamis

```
DPChange(money, coins)
begin
    MinNumCoins(0) ← 0
    for m ← 1 to money do
        MinNumCoins(m) ← ∞
        for i ← 1 to |coins| do
            if m ≥ coin_i then
                if MinNumCoins(m - coin_i) + 1 < MinNumCoins(m) then
                    MinNumCoins(m) ← MinNumCoins(m - coin_i) + 1
                end
            end
        end
    end
    return MinNumCoins(money)
end
```

“Programming” dalam “Dynamic Programming” tidak ada hubungannya dengan bahasa pemrograman!

Aplikasi program dinamis

- *Sequence alignment* dalam bioinformatika
- *Viterbi algorithm* untuk *hidden Markov models*
- Metode *value iteration* dalam *reinforcement learning*

Pertemuan berikutnya

- Uniform Cost Search
- Informed Search
- A*

Beberapa materi dari salindia ini diadaptasi dari Caltech CS154 dan Stanford CS221.

Terima kasih