Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Solusi Ujian Tengah Semester **IF2211 Strategi Algoritma**
Senin, 14 April 2025
Waktu: 100 menit
Dosen: Nur Ulfa Maulidevi (K1), Rinaldi (K2), Monterico Adrian (K3)

Berdoalah terlebih dahulu agar Anda berhasil dalam mengerjakan ujian ini!

A. Algoritma *Greedy*

1. Di sebuah komputer, terdapat banyak prosesor untuk mengerjakan *task*. Setiap *task* memiliki waktu mulai *T* dan lama eksekusi *D*. Satu prosesor dapat menjalankan banyak *task* secara sekuensial asalkan waktunya tidak beririsan ($T_i + D_i \leq T_{i+1}$). Jika terdapat 7 buah *task* (T_i, D_i) = [(2, 3), (0, 4), (8, 4), (5, 6), (9, 3), (8, 1), (12, 2)], dengan algoritma *greedy* tentukan dan jelaskan: **(7,5 + 7,5 = 15)**

(a) Berapa jumlah minimal prosesor yang dibutuhkan?

(b) Jika *task* yang memiliki $D > 4$ dapat dipecah menjadi dua *task* sehingga bisa dieksekusi secara paralel (beda prosesor) dengan ketentuan (T_i, D_i) → [(T_i, D_i div 2), (T_i, D_i div 2)], apakah hasilnya akan lebih optimal?

Jawaban:

Langkah-langkah:

1. Urutkan task dalam urutan menaik berdasarkan waktu mulainya (T) dan durasinya (D).
2. Mulai dengan prosesor p = 1
3. Masukkan task ke dalam prosesor p yang waktu mulainya lebih besar atau sama dengan waktu selesai task yang telah dipilih sebelumnya.
4. Jika masih ada task yang tersisa, tambahkan prosesor baru (p = p + 1), lalu ulangi langkah 3 sampai seluruh task sudah dikerjakan oleh prosesor

(a)

Diurutkan: (0, 4), (2, 3), (5, 6), (8, 1), (8, 4), (9, 3), (12, 2)

prosesor p = 1 : [(0, 4), (5, 6), (12, 2)]

prosesor p = 2 : [(2, 3), (8, 1), (9, 3)]

prosesor p = 3 : [(8, 4)]

Jadi dibutuhkan 3 prosesor

(b)

Task yang D > 4 hanya (5, 6), sehingga dipecah menjadi (5, 3) dan (5, 3)

Diurutkan: (0, 4), (2, 3), (5, 3), (5, 3), (8, 1), (8, 4), (9, 3), (12, 2)

prosesor p = 1 : [(0, 4), (5, 3), (8, 1), (9, 3), (12, 2)]

prosesor p = 2 : [(2, 3), (5, 3), (8, 4)]

Jadi dibutuhkan 2 prosesor, lebih optimal

B. *Divide and Conquer*

2. Diberikan sebuah larik **tidak terurut** yang berisi $n \geq 2$ bilangan bulat, misalnya $[x(1), \ldots, x(n)]$, asumsikan n perpangkatan 2, yaitu $n = 2^k$. Kita ingin menemukan jarak (*step*) terbesar d, yang didefinisikan sebagai nilai maksimum $x(j) - x(i)$ untuk **semua $j > i$.** Misalnya, untuk **(5 + 10 + 5 = 20)**

x = [22, 2, 8, 7, 4, 10, -1, 6], maka $d = x(6) - x(2) = 10 - 2 = 8$

(a) Jika diselesaikan dengan algoritma *brute force*, bagaimana langkah-langkahnya, dan berapa kompleksitas waktunya dalam notasi O-besar?

(b) Jika diselesaikan dengan algoritma *divide and conquer*, tuliskan langkah-langkahnya (boleh dalam notasi *pseudo-code* yang memperlihatkan tahap *divide*, *conquer*, dan *combine*), tentukan $T(n)$ dalam relasi rekurens, lalu tentukan notasi O-besarnya

(c) Perlihatkan proses *divide and conquer* untuk menemukan step terbesar pada contoh larik x di atas

Jawaban:

(a) Asumsikan *stepmax* diisi dengan nilai yang sangat besar

```
Loop mulai dari elemen ke-i = 1, 2, …, n – 1
    Loop mulai dari elemen ke-j = i+1, i+2, …, n
        Jika x(j) – x(i) > stepmax maka
            Stepmax = x(j) – x(i)
```

$T(n) = n(n + 1)/2 = O(n^2)$

Catatan: algoritma brute force dengan kompleksitas waktu O(n) juga dapat dibuat

(b) Divide: bagi larik menjadi 2 bagian, masing-masing berukuran n/2

Conquer: - cari stepmax pada upalarik kiri (*stepamaxkiri*)
- cari stepmax pada upalrik kanan (*stepmaxkanan*)

Combine: - temukan nilai minimun (*maks*) pada upalarik kiri
- temukan nilai maksimum (*min*) pada upalarik kanan
- hitung *stepmaxcross = maks – min*
- tentukan yang terbesar: max(*stepmaxkiri*, *stepmaxkanan*, *stepmaxcross*)

Pseudo-code:

```
procedure maxStepDC(x, i, j, stepmax, minimum, maksimum)
Algoritma:
  if j = i +1    { ukuran larik 2 elemen }
     stepmax ← x(j) – x(i)
     minimum ← x(i)
     maximum ← x(j)
  else           { ukuran larik > 2 elemen }
     mid ← (i + j) / 2
     maxStepDC(x, i, mid, stepmaxkiri, minL, maxL)
     maxStepDC(x, mid+1, j, stepmaxkanan2, minR, maxR)
     crossstep ← maxR – minL
     stepmax ← MAX(stepmaxkiri, stepmaxkanan, crossstep)
     minimum ← MIN(minL, minR)
     maksimum ← MAX(maxL, maxR)
  endif
```

Pemanggilan pertama kali: maxStepDC(x, 1, n, stepmax, minimum, maksimum)

Kompleksitas algoritma, diihitung dari jumlah operasi perbandingan:

$$T(n) = \begin{cases} a & ,n = 2 \\ 2T\left(\frac{n}{2}\right) + c & ,n > 2 \end{cases}$$

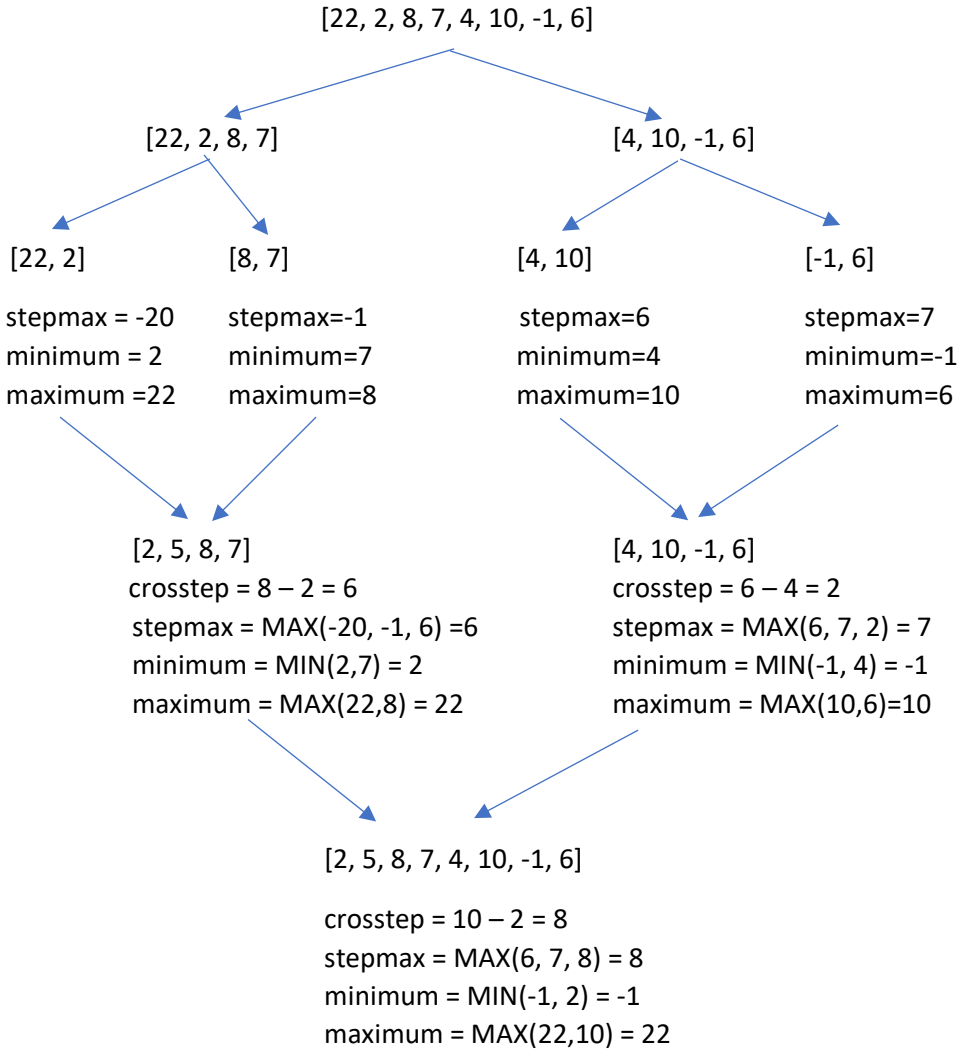
Dengan menggunakan Teorema Master,

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d$$

$$T(n) \text{ adalah } \begin{cases} O(n^d) & ,a < b^d \\ O(n^d \log n) & ,a = b^d \\ O(n^{\log_b a}) & a > b^d \end{cases}$$

a = 2, b = 2, d = 0 $\rightarrow$ $2 > 2^0 \rightarrow$ Case 3: $O(n^{log_2 2}) = O(n)$

(c)



3. Misalkan Anda akan memilih satu di antara 3 algoritma berikut:
   (i) Algoritma A memecahkan persoalan berukuran *n* dengan membaginya menjadi lima upa-persoalan berukuran *n*/2, menyelesaikan setiap upa-persoalan secara rekursif, lalu menggabungkan solusinya dalam waktu linier.
   (ii) Algoritma B memecahkan persoalan berukuran *n* dengan membaginya menjadi dua upa-persoalan masing-masing berukuran *n* – 1, menyelesaikan setiap upa-persoalan secara rekursif, lalu menggabungkan solusinya dalam waktu konstan.

(iii) Algoritma C memecahkan persoalan berukuran *n* dengan membaginya menjadi sembilan upa-persoalan berukuran *n*/3, memecahkan setiap upa-persoalan secara rekursif, lalu menggabungkan solusinya dalam waktu kuadratik.

Tentukan kompeksitas waktu, *T*(*n*), untuk setiap algoritma, dan kompleksitas waktu asimptotiknya dalam notas O-besar, lalu tentukan algoritma mana yang akan Anda pilih dan jelaskan alasannya. **(4 + 4 + 4 + 3 = 15)**

Jawaban:

Teorema Master: $T(n) = aT\left(\frac{n}{b}\right) + cn^d$

$$T(n) \text{ adalah } \begin{cases} O(n^d) & , a < b^d \\ O(n^d \log n) & , a = b^d \\ O(n^{\log_b a}) & a > b^d \end{cases}$$

(i) $T(n) = 5T\left(\frac{n}{2}\right) + cn \rightarrow a = 5, b = 2, d = 1 \rightarrow 5 > 2^1 \rightarrow$ Case 3: $O(n^{\log_2 5}) = O(n^{2,232})$

(ii) $T(n) = 2T(n-1) + c \rightarrow$ tidak bisa pakai Teorema Master, pakai cara iteratif

T(n) = 2T(n – 1) + c

= 2(2T(n – 2) + c) + c = 4T(n – 2) + 2c + c

= 4(2T(n – 3) + c) + 3c

= 8 T(n – 3) + 4c + 3c = 8(T(n – 3) + 7c

= $2^k T(n-k) + (2^k - 1)$c

Ambil k = n, maka T(n) = $2^n T(0) + (2^n - 1)$c

$\rightarrow$ misalkan T(0) = a $\rightarrow$ T(n) = $2^n + (2^n - 1)$c = $O(2^n)$

(iii) $T(n) = 9T\left(\frac{n}{3}\right) + cn^2$ $a = 9, b = 3, d = 2 \rightarrow 9 = 3^2 \rightarrow$ Case 2: $O(n^2 \log n)$

Spektrum kompeksitas:

$n^2 \log n < n^{2,232} < 2^n$

Jadi, algoritma yang dipilih adalah algoritma C karena kompleksitas waktunya paling kecil

C. Decrease and Conquer

4. Terdapat sebuah timbangan seperti pada gambar di samping. Timbangan digunakan untuk menentukan satu buah bola dari 9 buah bola, yang memiliki **bobot lebih berat** dibandingkan yang lain. Delapan buah bola yang lain memiliki bobot yang sama. Dengan pendekatan *Decrease and Conquer*, **(7,5 + 7,5 = 15)**

   (a) Tentukan langkah-langkah penimbangan sebanyak **dua kali** untuk mendapatkan satu buah bola yang lebih berat dibandingkan bola yang lain. Tentukan termasuk pendekatan *Decrease and Conquer* yang mana (*decrease by constant*, *decrease by constant factor*, atau *decrease by variable size*).

   (b) Jika *n* merepresentasikan banyaknya bola (dalam kelipatan 3), dan (*n* - 1) adalah banyaknya bola dengan bobot yang sama, tentukan kompleksitas waktu pendekatan *Decrease and Conquer* pada jawaban (a); kemudian tentukan kompleksitasnya dalam notasi Big-O.

Jawaban:

a. Langkah-langkah decrease by constant factor:

1. Bagi 9 buah bola menjadi 3 bagian @ 3 bola (misal bagian A, B, dan C).
2. Timbang bagian A dan bagian B, sedangkan bagian C simpan di tanah.
3. Ambil satu bagian yang paling berat dari hasil penimbangan (jika timbangan A dan B sama, berarti yang paling berat bagian C).

4. Bagi lagi menjadi 3 bagian untuk satu bagian hasil langkah (3) yang artinya tiap bagian hanya berisi 1 bola, dan lakukan penimbangan untuk 2 bagian saja. Bagian terberat dari hasil penimbangan ke-2 inilah bola yang paling berat diantara yang lain.

b. T(n) = T(n/3) + 1 → tiap kali membagi 3 dan hanya 1 yang diproses, dengan 1 kali operasi perbandingan.

Teorema Master: $T(n) = aT\left(\frac{n}{b}\right) + cn^d$

$$T(n) \text{ adalah } \begin{cases} O(n^d) & , a < b^d \\ O(n^d \log n) & , a = b^d \\ O(n^{\log_b a}) & a > b^d \end{cases}$$

Dengan teorema master: a=1, b=3, c=1, d=0 → $a = b^d$ → $O(\log_3 n)$ atau O(log n)

D. DFS, BFS, IDS

5. Terdapat sebuah informasi berupa senarai ketetanggaan sebagai berikut, untuk merepresentasikan suatu graf berarah.

{ A : B,D,
B : C, F,
C : E, G, H,
D : F,
E : B, F,
F : A
H : A }

(5 + 5 + 5 + 5 = 20)

(a) Gambarkan graf berarah sesuai dengan informasi senarai ketetanggaan tersebut.

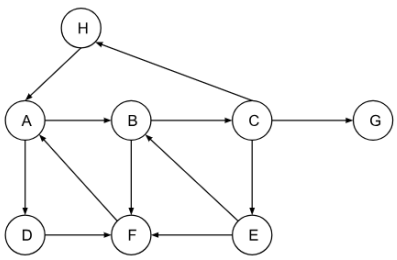
(b) Jika dari simpul A ingin menuju simpul F, **tentukan jalur penelusuran/ pencarian dengan BFS**, dan tentukan **jalur hasil untuk menuju F dari simpul A**. Jika terdapat lebih dari satu simpul tetangga, pemilihan simpul sesuai urutan abjad.

(c) Jika dari simpul A ingin menuju simpul F, **tentukan jalur penelusuran/ pencarian dengan DFS**, dan tentukan **jalur hasil untuk menuju F dari simpul A**. Jika terdapat lebih dari satu simpul tetangga, pemilihan simpul sesuai urutan abjad.

(d) Jika dari simpul A ingin menuju simpul F, **tentukan jalur penelusuran/ pencarian dengan DLS** dengan batas kedalaman **3**, dan tentukan **jalur hasil untuk menuju F dari simpul A**. Jika terdapat lebih dari satu simpul tetangga, pemilihan simpul sesuai urutan abjad.

Jawaban:

a. Gambar Graf Berarah



b. BFS

Jalur penelusuran: A, B, D, C, F. Jika jalur penelusuran dituliskan dalam bentuk pohon, harus jelas urutan pembangkitan simpulnya. Jalur hasil: A - B - F.

c. DFS

Jalur penelusuran: A, B, C, E, F. Jika jalur penelusuran dituliskan dalam bentuk pohon, harus jelas urutan pembangkitan simpulnya. Jalur hasil: A - B - C - E - F.

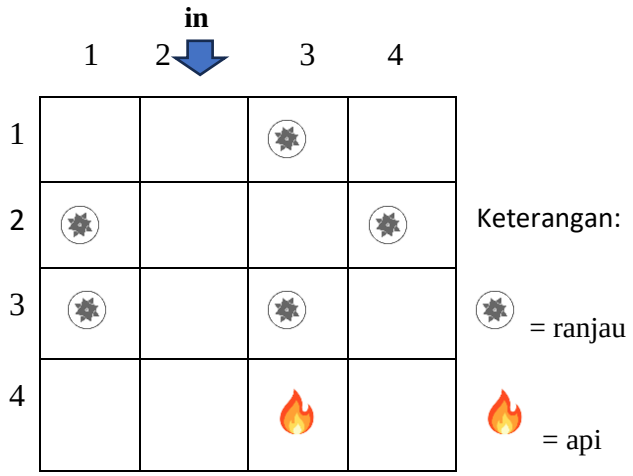
d. DLS; catatan → bedakan DLS dan IDS, tidak ada iterasi penambahan kedalaman dalam DLS. DLS adalah DFS yang kedalamannya dibatasi.
Jalur penelusuran: A, B, C, E, G, H, F. Jika jalur penelusuran dituliskan dalam bentuk pohon, harus jelas urutan pembangkitan simpulnya. Jalur hasil: A, B, F.

6. Sebuah robot pemadam api memasuki sebuah matriks 4 x 4 (indeks matriks dimulai dari 1). Robot hanya dapat bergerak satu sel ke kiri, ke kanan, ke bawah, atau ke atas. Di dalam matriks terdapat beberapa sel ranjau (robot tidak tahu di dalam sel ada ranjau). Jika robot memasuki sel ranjau, makai robot mati karena ledakan ranjau. Robot mulai berjalan dari sebuah titik masuk (**in**) sampai menemukan sel api tersebut. Urutan gerakan robot dari setiap sel harus konsisten, yaitu kiri, kanan, atas, bawah.

(a) Gambarkan pohon ruang status lengkap sebelum pencarian

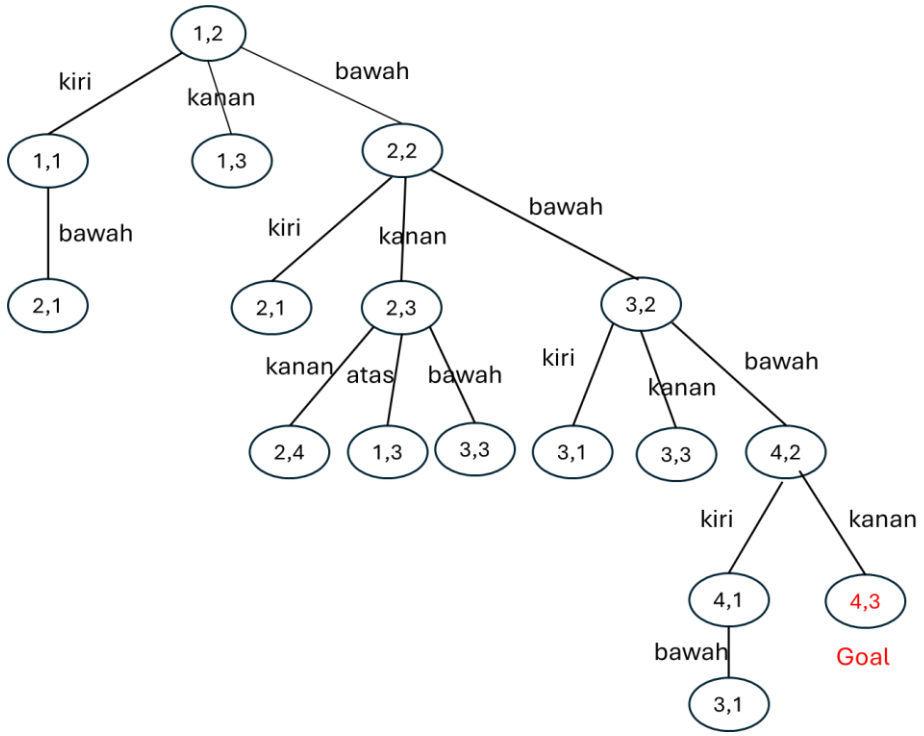
(b) Gambarkan pohon ruang status pencarian sel yang mengandung api secara DFS. Beri nomor setiap status sesuai urutan pembangkitannya, setiap sisi (cabang) pada pohon ditulis gerakannya (kiri, kanan, atas, bawah). Pada setiap status tulis koordinat elemen matriks yang dimasuki.

(c) Ulangi soal (b) untuk BFS **(5 + 5 + 5 = 15)**

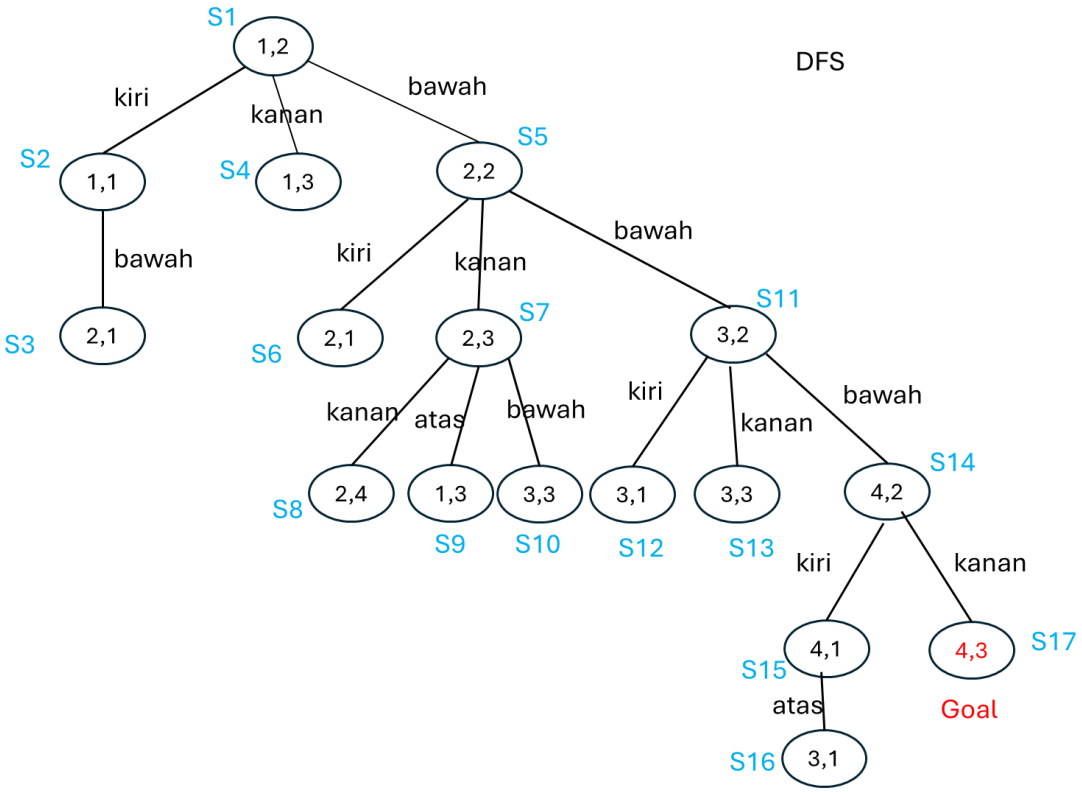


Jawaban:

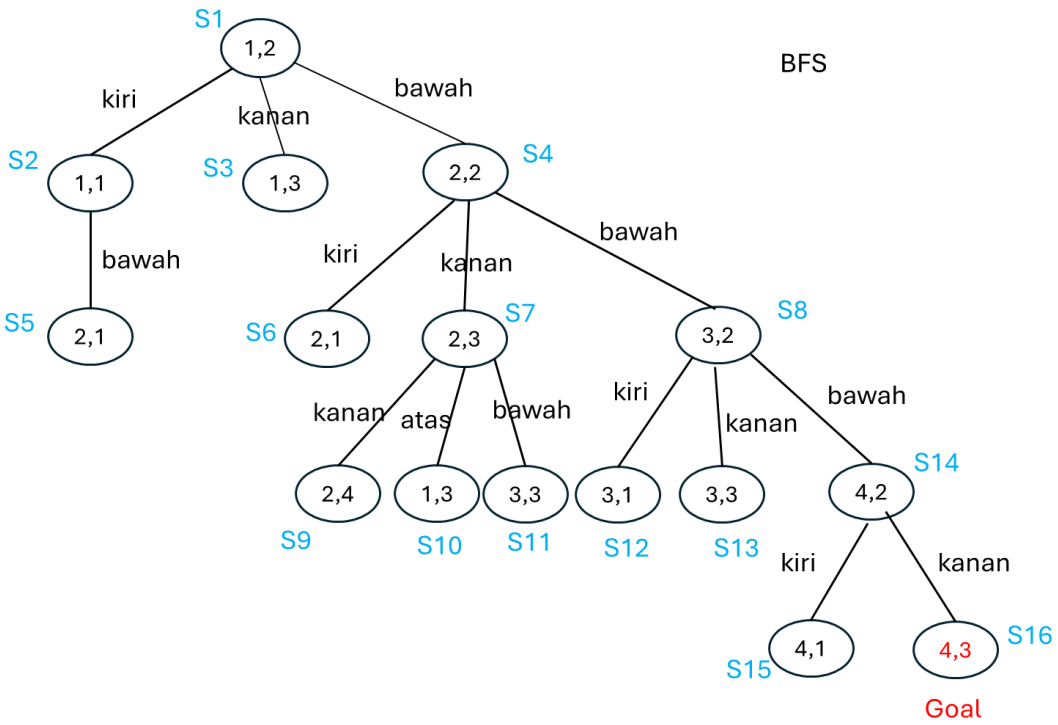
(a)



(b)


DFS
S1
1,2
kiri
kanan
bawah
S2
1,1
S4
1,3
S5
2,2
bawah
kiri
kanan
bawah
S3
2,1
S6
2,1
S7
2,3
S11
3,2
kanan
atas
bawah
kiri
kanan
bawah
S8
2,4
1,3
3,3
3,1
3,3
4,2
S14
S9
S10
S12
S13
kiri
kanan
4,1
4,3
S17
S15
atas
Goal
S16
3,1

(c)


BFS
S1
1,2
kiri
kanan
bawah
S2
1,1
S3
1,3
S4
2,2
bawah
kiri
kanan
bawah
S5
2,1
S6
2,1
S7
2,3
S8
3,2
kanan
atas
bawah
kiri
kanan
bawah
S9
2,4
1,3
3,3
3,1
3,3
4,2
S14
S10
S11
S12
S13
kiri
kanan
4,1
4,3
S16
S15
Goal