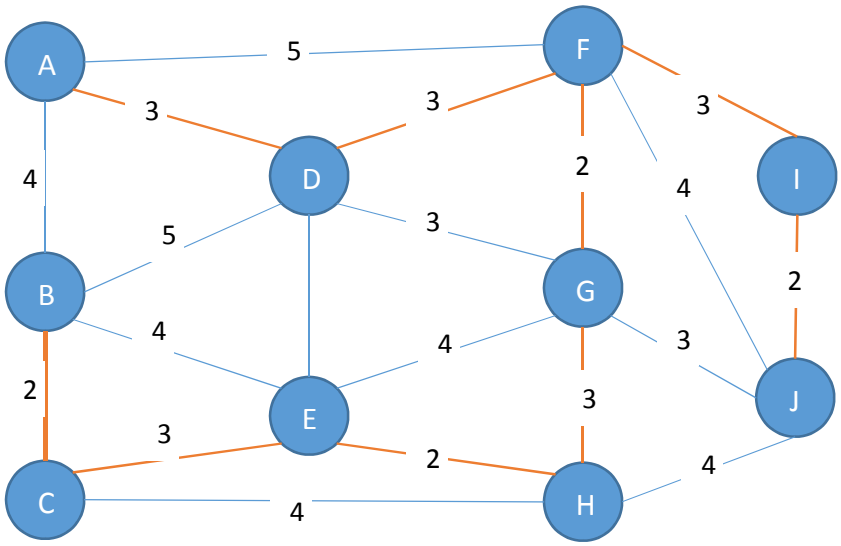
Solusi UTS Stima

1. a. (Nilai 5) Representasikanlah gambar kota di atas menjadi sebuah graf, dengan simpul merepresentasikan rumah, dan bobot sisi merepresentasikan jumlah paving block yang dibutuhkan. Jembatan dihitung sebagai satu paving block.



b. (Nilai 15) Jika diselesaikan dengan exhaustive search, jelaskan strateginya seperti apa (tidak perlu pseudo-code), lalu tentukanlah kompleksitas algoritmanya dalam notasi big O.

Persoalan ini adalah persoalan minimum spanning tree.

Alternatif 1 strategi:

- enumerasi semua spanning tree berupa lintasan hamilton dgn cara membuat semua permutasi semua simpul (telah memenuhi syarat tidak siklik), terdapat (n-1)!
- evaluasi setiap spanning tree dengan menghitung bobot lintasan
- Ambil spanning tree dengan bobot paling minimum

Kompleksitas: O(n.n!), n adalah jumlah simpul

Alternatif 2 strategi:

- enumerasi semua subset sisi, terdapat 2^n, dengan n adalah jumlah sisi.
- evaluasi setiap subset yang memenuhi spanning tree (tidak siklik, melewati semua simpul) dengan menghitung bobot lintasan
- Ambil spanning tree dengan bobot paling minimum

Kompleksitas: O(n.2^n), n adalah jumlah sisi

2. (Nilai 15) Jika diselesaikan dengan greedy, rancanglah strategi greedy terbaiknya, dan tentukanlah kompleksitas algoritmanya dalam notasi big O.

Alternatif 1: dgn menggunakan Kruskal:

1. Sort semua sisi berdasarkan jumlah paving block: BC(2), EH(2), FG(2), IJ(2), AD(3),…
2. Tambahkan sisi secara berurutan untuk membentuk pohon jika tidak membentuk siklik. Jika membentuk siklik, sisi diabaikan. Hal ini dilakukan sampai semua simpul sudah dicapai.

Solusi optimum : BC(2)-EH(2)-FG(2)-IJ(2)-AD(3)-CE(3)-DF(3)-FI(3)-GH(3). Total bobot=23.

Kompleksitas: O(|E|log|E|)

Alternatif 2: dgn menggunakan Prim:

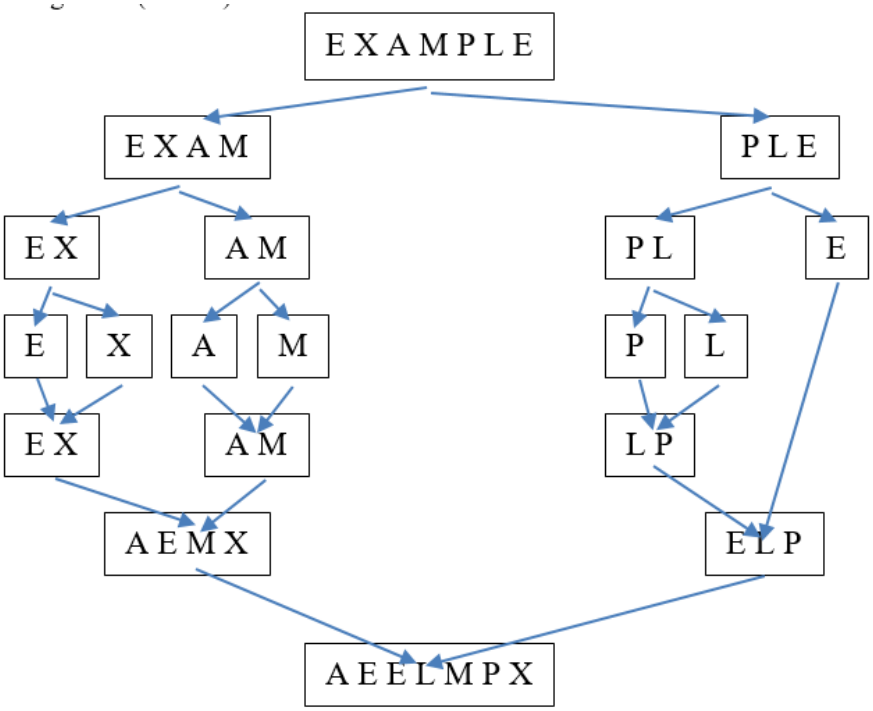
Ambil simpul awal (misalnya A atau simpul dgn bobot terpendek), lalu bangun tree T secara greedy, dengan menambahkan sisi terpendek yang menghubungkan salah satu simpul yang telah terpilih dengan simpul yang belum terpilih.

Solusi optimum: A-D-F-G-H-E-C-B-I-J atau B-C-E-H-G-F-D-A-I-J. Total bobot=23.

Kompleksitas: $O(n^2)$

3. Jawaban untuk MergeSort (Nilai 7)

Banyaknya prosedur MergeSort yang dipanggil, tidak dinilai. Selama proses benar, maka mendapatkan nilai penuh.
Partisi yang dilakukan harus konsisten: ((i+j div 2) atau ((i+j div 2)-1)).
Jika mahasiswa menggunakan basis 2, maka harus menyertakan “pseudo code” untuk penanganan basis 2, karena basis 2 harus ada pemeriksaan nilai antara 2 elemen tersebut. Jika tidak ada “pseudo code” ketika menggunakan basis 2, nilai dikurangi 1.



Pemanggilan prosedur MergeSort:

MergeSort (A,1,7)

a. MergeSort (A,1,4)
   i. MergeSort(A,1,2)
      1. MergeSort(A,1,1)
      2. MergeSort(A,2,2)
   ii. MergeSort(A,3,4))
      1. MergeSort(A,3,3)
      2. MergeSort(A,4,4)

b. MergeSort(A,5,7)
   i. MergeSort(A,5,6)
      1. MergeSort(A,5,5)
      2. MergeSort(A,6,6)
   ii. MergeSort(A,7,7)

Jawaban untuk QuickSort (Nilai 8)

Banyaknya prosedur QuickSort yang dipanggil, tidak dinilai. Selama proses benar, maka mendapatkan nilai penuh.

Dalam kuliah diberikan 2 versi prosedur QuickSort, yaitu versi pivot disertakan dalam partisi (versi 1), dan versi pivot tidak disertakan dalam partisi (versi 2). Jika mahasiswa menggunakan versi "campur-campur", harus menyertakan "pseudo code" agar jelas proses QuickSort yang dilakukan. Jika tidak, akan sulit diperiksa apakah prosedur QuickSort yang dilakukan konsisten atau tidak.

Untuk kedua versi, swap dilakukan ketika:

- A[p] ≥ pivot; dan
- A[q] ≤ pivot; dan
- p < q;

dengan p adalah variabel yang bergerak dari kiri ke kanan, dan q adalah variabel yang bergerak dari kanan ke kiri. Setelah swap, p dimulai lagi dari p=p+1, dan q dimulai dari q=q-1.

Dalam prosedur QuickSort tidak ada "merge", karena selesai melakukan partisi, array sudah dalam posisi terurut.

Versi 1, pivot masuk ke partisi, p dimulai dari elemen pertama.

p=1, q=7

Iterasi	1	2	3	4	5	6	7
	E	X	A	M	P	L	E
1	P						q

Tukar E di indeks 1 dengan E di indeks 7, A[p] ≥ pivot, dan A[q] ≤ pivot

Mulai lagi penelusuran dengan indeks p= p+1=2, q=q-1=6, belum dipartisi, kondisi larik setelah di iterasi:

E	X	A	M	P	L	E
	p	Q				

Tukar X di indeks 2 dengan A di indeks 3, karena A[p] ≥ pivot, dan A[q] ≤ pivot, kondisi larik menjadi:

E	A	X	M	P	L	X

Mulai lagi penelusuran dengan indeks p=p+1=3, dan q=q-1=2

Karena p > q maka penelusuran dihentikan, proses pemisahan dilakukan pada indeks q
Panggil prosedur (2) QuickSort(A,1,2) dan (3) QuickSort(A,3,7)

QuickSort(A,1,2)

2

E		A
p		q

Tukar E di indeks 1 dengan A di indeks 2, A[p] ≥ pivot, dan A[q] ≤ pivot
Mulai lagi penelusuran dengan indeks p= p+1=2, q=q-1=1
Karena p > q maka penelusuran dihentikan, proses pemisahan dilakukan pada indeks q
Panggil prosedur (4) QuickSort(A,1,1) dan (5) QuickSort(A,2,2). Karena hanya satu elemen, tidak melakukan apapun pada pemanggilan dua prosedur tersebut.
Kondisi larik setelah pemanggilan QuickSort(A,1,2):

A	E	X	M	P	L	E

3 QuickSort(A,3,7)

X	M	P	L	E
p				q

Tukar X di indeks 3 dengan E di indeks 7, karena A[p] ≥ pivot, dan A[q] ≤ pivot
Mulai lagi penelusuran dengan indeks p= p+1=4, q=q-1=6, belum dipartisi, kondisi larik setelah di iterasi:

E	M	P	L	X
			Q	p

Iterasi berhenti ketika p=7 (karena sudah indeks terakhir), dan q tetap di 6 karena A[q] ≤ pivot.
Penukaran tidak dilakukan karena p > q, jadi partisi dilakukan pada q = 6
Panggil prosedur (6) QuickSort(A,3,6) dan (7) QuickSort(A,7,7). Karena prosedur QuickSort(A,7,7) hanya 1 elemen, maka sudah mencapai basis.

6 QuickSort(A,3,6)

E	M	P	L
P			q

Awal iterasi p berada pada indeks 3 dan q berada pada indeks 6. Indeks p tidak maju karena A[p] ≥ pivot, sedangkan q akan terus berkurang hingga sampai pada indeks 3, karena A[q] ≤ pivot. Kemudian ditukar elemen pada indeks p dan q, namun karena mengacu pada hal yang sama, maka tetap. Partisi dilakukan pada indeks q, dan memanggil prosedur (8) QuickSort(A,3,3) dan (9) QuickSort(A,4,6).
Karena prosedur QuickSort(A,3,3) hanya terdiri atas 1 elemen maka sudah mencapai basis.

9 QuickSort(A,4,6)

M	P	L
p		q

Tukar M di indeks 4 dengan L di indeks 6, karena A[p] ≥ pivot, dan A[q] ≤ pivot
Mulai lagi penelusuran dengan indeks p= p+1=5, q=q-1=5.
Setelah penelusuran, p berhenti di indeks 5 karena A[p] ≥ pivot, dan q berhenti di indeks 4 karena A[q] ≤ pivot.
Karena p > q maka penelusuran dihentikan, proses pemisahan dilakukan pada indeks q.
Panggil prosedur (10) QuickSort(A,4,4) dan (11) QuickSort(A,5,6). Karena QuickSort(A,4,4) hanya satu elemen, maka sudah mencapai basis.

11 QuickSort(A,5,6)

P	M
p	q

Tukar P di indeks 5 dengan M di indeks 6, karena A[p] ≥ pivot, dan A[q] ≤ pivot
Mulai lagi penelusuran dengan indeks p= p+1=6, q=q-1=5.
Karena p > q maka penelusuran dihentikan, proses pemisahan dilakukan pada indeks q.
Panggil prosedur (12) QuickSort(A,5,5) dan (13) QuickSort(A,6,6). Karena keduanya hanya satu elemen, maka sudah mencapai basis.
Kondisi setelah pemanggilan terakhir QuickSort adalah sebagai berikut.

Hasil:	A	E	E	L	M	P	X

Versi 2, pivot tidak termasuk partisi manapun, pemeriksaan p dimulai setelah elemen pertama

Iterasi		1	2	3	4	5	6	7
		E	X	A	M	P	L	E
1		pivot	p					q

Tukar X di indeks 2 dengan E di indeks 7

Mulai lagi penelusuran dengan indeks p= p+1=3, q=q-1=6, belum dipartisi, kondisi larik:

E	E	A	M	P	L	X
pivot		q	p			

Karena p>q maka partisi dilakukan di sini, swap pivot dengan posisi terakhir q (A[1] dengan A[3])

Panggil prosedur:

QuickSort(A,E)

QuickSort(MPLX)

Posisi akhir pada tahap ini:

A	E	E	M	P	L	X

2	QuickSort(A, E)		
		A	E
		pivot	pq

q bergerak ke kiri, p berhenti di indeks 2, sehingga p=2, q=1.

Karena p>q maka partisi dilakukan di sini, swap A[q] dengan pivot (A dengan A), panggil prosedur:

QuickSort(E)

3	QuickSort(E)	
		E

Sudah basis

4	M	P	L	X
	pivot	p		q

Tukar P di indeks 5 dengan L di indeks 6

Mulai lagi penelusuran dengan indeks p=5+1=6, q=6-1=5

Karena p>q maka partisi dilakukan di sini, swap A[4] dengan A[5]

Panggil:

QuickSort(L)

QuickSort(P,X)

Posisi akhir pada tahap ini:

L	M	P	X

5	QuickSort(L)		
		L	

Sudah basis

6	QuickSort(P,X)		
		P	X
		pivot	pq

q bergerak ke kiri, p berhenti di indeks 7, sehingga p=7, q=6.

Karena p>q maka partisi dilakukan di sini, panggil prosedur:

QuickSort(X)

7	QuickSort(X)	
		X

Sudah basis

Hasil:	A	E	E	L	M	P	X

4. Jawaban untuk maximum sub-array (Terdapat sebuah larik A yang berisi bilangan bulat positif dan negatif sebagai berikut. {2, -4, 1, 9, -6, 7, -3}).
   a. (Nilai 10)
      - Bagi larik menjadi dua bagian dengan ukuran yang 'relatif' sama.
      - Cari nilai maksimum di antara ketiga nilai berikut ini:
        - Maksimum jumlah upalarik dari bagian kiri (pemanggilan rekursif)
        - Maksimum jumlah upalartik bagian kanan (pemanggilan rekursif)
        - Maksimum jumlah upalarik yang melewati bagian tengah larik (mengandung bagian kiri dan kanan), dengan cara:
          - Cari jumlah maksimum upalarik mulai dari elemen tengah ke kiri
          - Cari jumlah maksimum upalarik mulai dari elemen tengah+1 ke kanan
          - Kombinasikan keduanya dan jumlahkan hasilnya

   b. (Nilai 5)
      Kasus basis: terdiri atas 1 elemen, mengembalikan elemen itu sendiri.
      Kasus rekurens: 2 kali pemanggilan prosedur dengan parameter setengahnya, ditambah waktu untuk mencari jumlah upalarik yang melewati bagian tengah (waktu linear).

$$T(n) = \begin{cases} a & ,n = 1 \\ 2T(n/2) + cn & ,n > 1 \end{cases}$$

   c. (Nilai 5)
      Dari formula di butir (4b), diketahui nilai a = 2, b = 2, dan d = 1. Berdasarkan teorema master, karena $a = b^d$ maka kompleksitas pendekatan Divide and Conquer untuk kasus (a) adalah:
      O(n log n)

5. Diketahui n = 9 (jumlah data)
   s = Bilangan terbesar ke-5 = bilangan terkecil ke $\lceil 9/2 \rceil$ = median
   Partisi latrik dengan metode partisi di dalam algoritma Quicksort varian 2

13	9	18	6	8	11	15	7	12
		↑p						↑q

Pertukarkan 18 ↔ 12, lalu pindai dari p+1 dan q-1

13	9	12	6	8	11	15	7	18
						↑p	↑q	

Pertukarkan 15 ↔ 7, lalu pindai dari p+1 dan q-1

13	9	12	6	8	11	7	15	18
						↑q	↑p	(p > q stop)

7	9	12	6	8	11	**13**	15	18

Karena s = 7 > 5 maka ambil larik bagian kiri

7	9	12	6	8	11

↑p ↑q

Pertukarkan 9 ↔ 8, lalu pindai dari p+1 dan q-1

7 6 12 9 8 11

↑q ↑p (p > q stop)

6 **7** 12 9 8 11

Karena s = 2 < 5 maka ambil larik bagian kanan

12 9 8 11

↑q

11 9 8 **12**

Karena s = 6 > 5 maka ambil larik bagian kiri

11 9 8

↑q

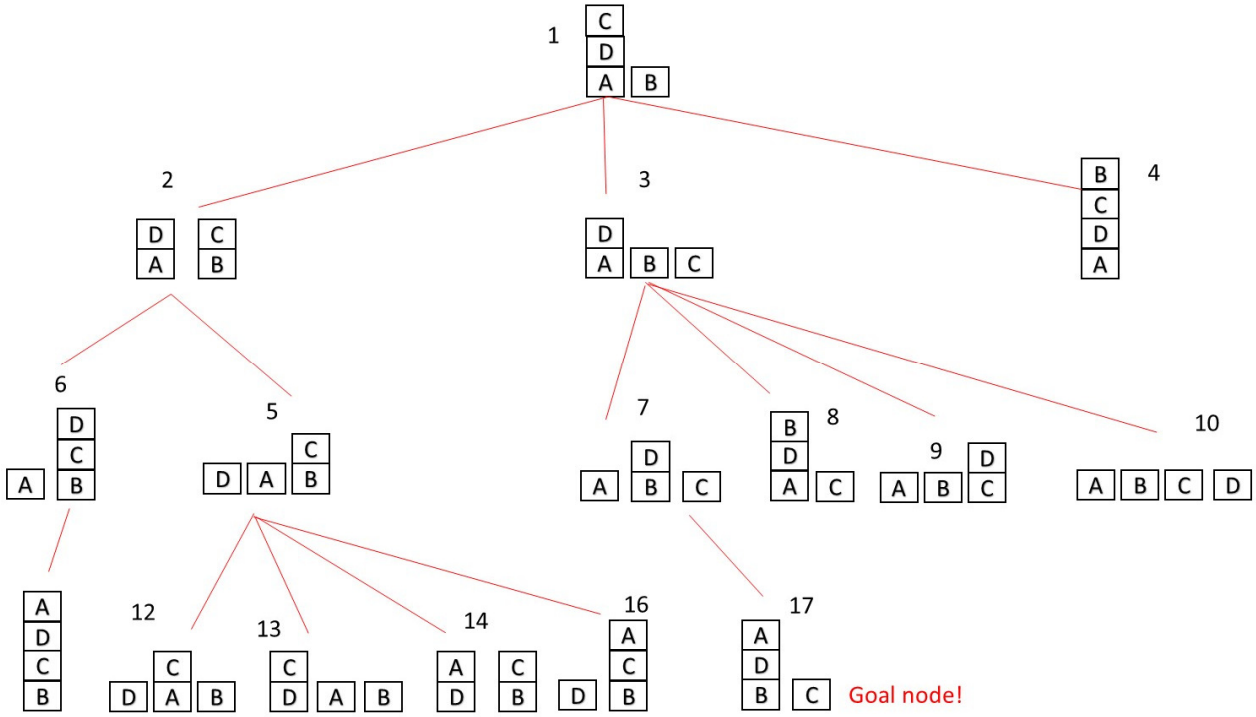
9 8 11

↑q

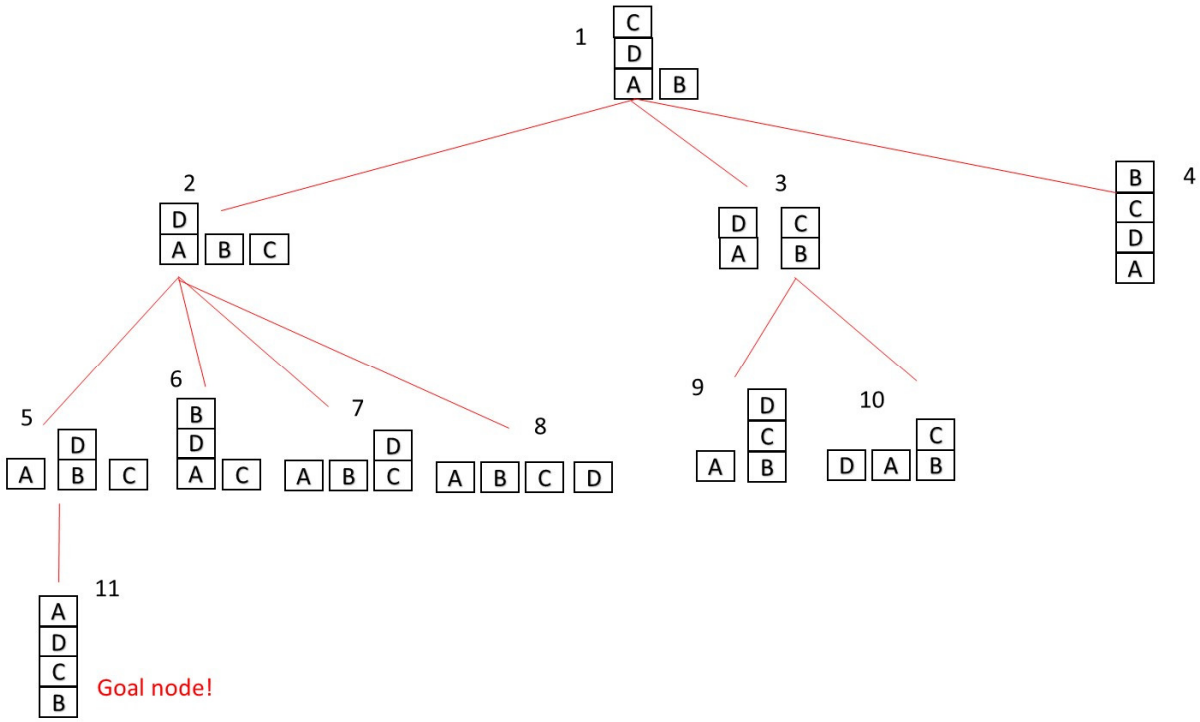
Karena s = 5 maka nilai terbesar ke-5 ditemukan, yaitu 11

6. BFS

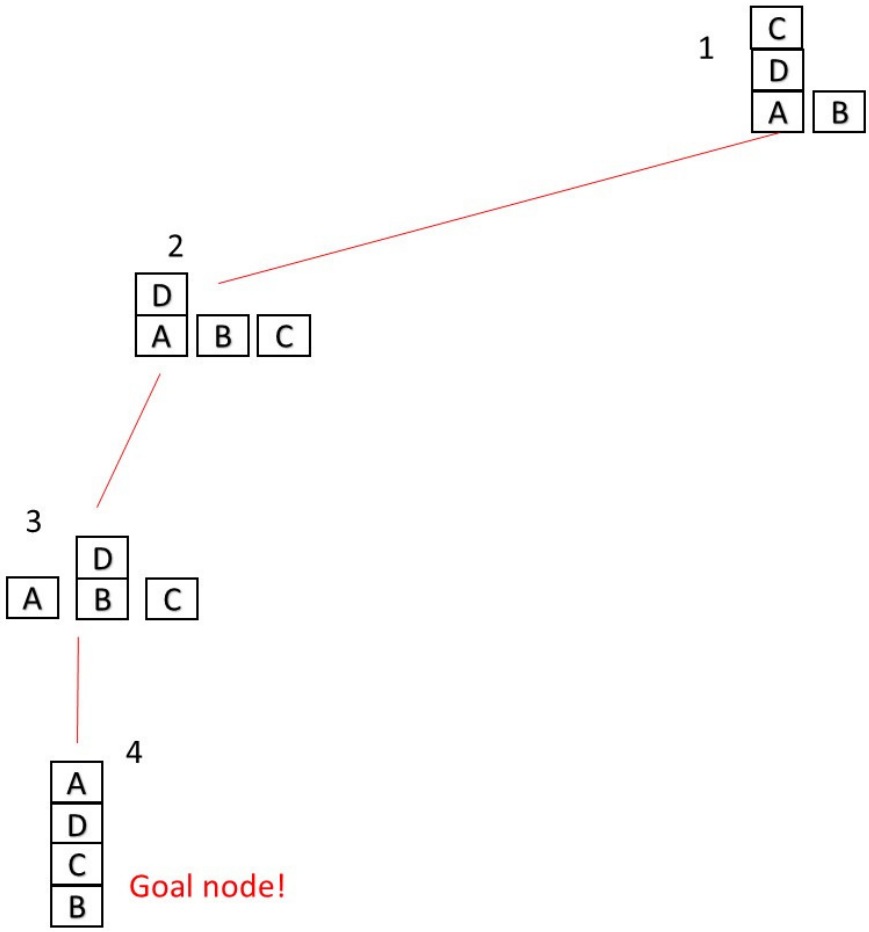
Salah satu kemungkinan solusi:



Kemungkinan lain:



DFS:
Salah satu kemungkinan solusi


1
C
D
A
B
2
D
A
B
C
3
D
A
B
C
4
A
D
C
B
Goal node!