Fakultas Informatika
School of Computing
Telkom University

CII2K3
STRATEGI ALGORITMA

Topik 5:
BACKTRACKING

1. Penjelasan umum
2. Permasalahan n-Queen
3. Pencarian sirkuit Hamilton
4. Pewarnaan Graf
5. Permasalahan jumlah nilai subhimpunan

Penjelasan umum

Penjelasan umum

- Sejauh ini kita telah mempelajari strategi algoritma Brute Force, Greedy, Divide and Conquer, dan Dynamic Programming
- Greedy, Divide and Conquer, dan Dynamic Programming sering kali memberikan algoritma yang lebih efisien dibandingkan dengan Brute Force, namun masih banyak masalah yang tidak dapat diselesaikan secara akurat oleh ketiganya
- Backtracking adalah strategi algoritma yang merupakan perbaikan dari *exhaustive search*

Ide dasar

- Mengkonstruksi solusi per komponen dan mengevaluasi solusi parsial yang didapatkan sbb:
  - Jika solusi parsial tersebut dapat dilanjutkan tanpa melanggar *constraint* yang ada, maka kita tambahkan komponen lainnya yang tidak melanggar *constraint*
  - Jika solusi partial yang didapatkan tidak dapat dilanjutkan (i.e. penambahan komponen apapun akan menyebabkan pelanggaran *constraint)*, maka ganti komponen terakhir dengan komponen lainnya

Ide dasar

- Mengkonstruksi solusi per komponen dan mengevaluasi solusi parsial yang didapatkan sbb:
  - Jika solusi parsial tersebut dapat dilanjutkan tanpa melanggar *constraint* yang ada, maka kita tambahkan komponen lainnya yang tidak melanggar *constraint*
  - Jika solusi partial yang didapatkan tidak dapat dilanjutkan (i.e. penambahan komponen apapun akan menyebabkan pelanggaran *constraint)*, maka ganti komponen terakhir dengan komponen lainnya
- Umumnya, diilustrasikan oleh state-space tree

State-space tree

- Merupakan graf pohon, dimana:
  - Akar merepresentasikan tahap awal (*initial state* sebelum konstruksi solusi dimulai
  - Simpul-simpul pada level ke-i merepresentasikan pilihan untuk dipilih sebagai komponen ke-i
  - Daun merepresentasikan ujung dari konstruksi solusi yang tidak menjanjikan atau ujung dari konstruksi solusi yang lengkap

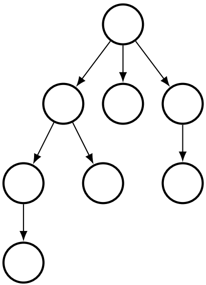
State-space tree

- Merupakan graf pohon, dimana:
  - Akar merepresentasikan tahap awal (*initial state* sebelum konstruksi solusi dimulai
  - Simpul-simpul pada level ke-i merepresentasikan pilihan untuk dipilih sebagai komponen ke-i
  - Daun merepresentasikan ujung dari konstruksi solusi yang tidak menjanjikan atau ujung dari konstruksi solusi yang lengkap
- Simpul pada *state-space tree* dikatakan menjanjikan jika dia termasuk ke dalam solusi partial yang masih mungkin dilanjutkan untuk mendapatkan solusi lengkap

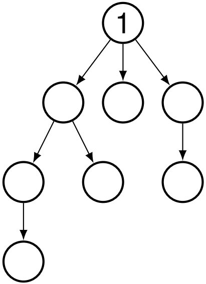
State-space tree

- Merupakan graf pohon, dimana:
  - Akar merepresentasikan tahap awal (*initial state* sebelum konstruksi solusi dimulai
  - Simpul-simpul pada level ke-i merepresentasikan pilihan untuk dipilih sebagai komponen ke-i
  - Daun merepresentasikan ujung dari konstruksi solusi yang tidak menjanjikan atau ujung dari konstruksi solusi yang lengkap
- Simpul pada *state-space tree* dikatakan menjanjikan jika dia termasuk ke dalam solusi partial yang masih mungkin dilanjutkan untuk mendapatkan solusi lengkap
- Umumnya, *state-space tree* dikonstruksi secara *depth-first search* (DFS)

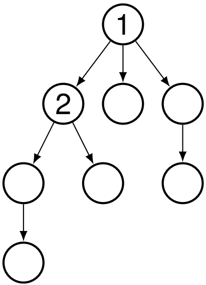
Recall: Depth First Search (DFS)



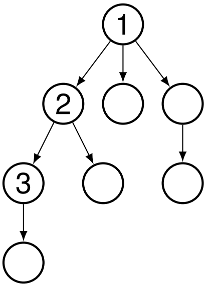
Recall: Depth First Search (DFS)



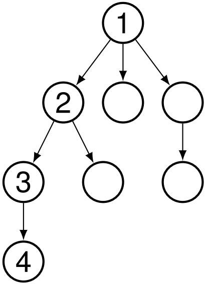
Recall: Depth First Search (DFS)



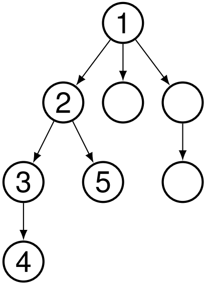
Recall: Depth First Search (DFS)



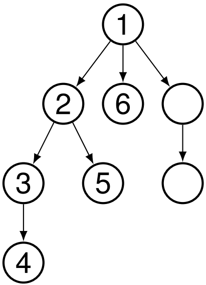
Recall: Depth First Search (DFS)



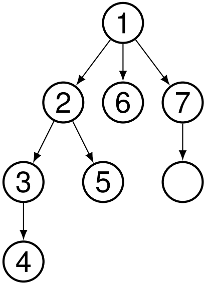
Recall: Depth First Search (DFS)



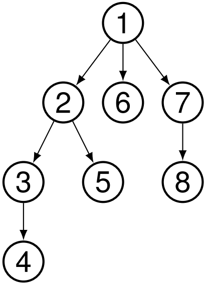
Recall: Depth First Search (DFS)



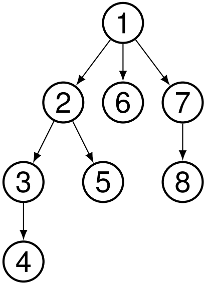
Recall: Depth First Search (DFS)



Recall: Depth First Search (DFS)

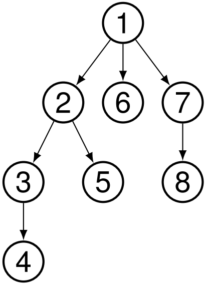


Recall: Depth First Search (DFS)

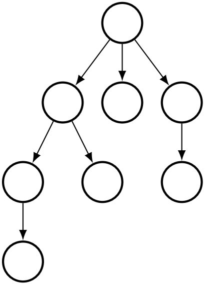


kiri ke kanan

Recall: Depth First Search (DFS)

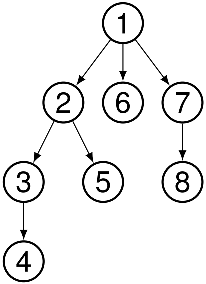


kiri ke kanan

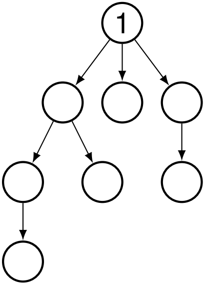


kanan ke kiri

Recall: Depth First Search (DFS)

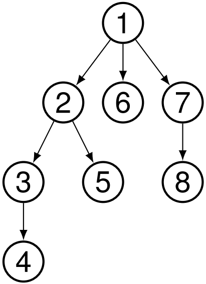


kiri ke kanan

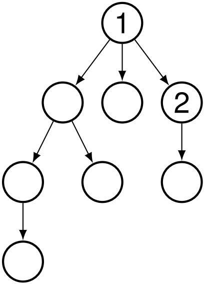


kanan ke kiri

Recall: Depth First Search (DFS)

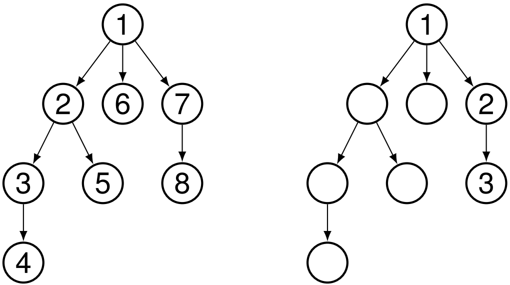


kiri ke kanan



kanan ke kiri

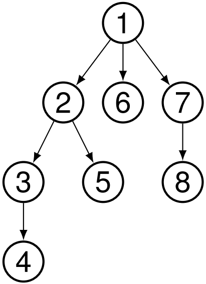
Recall: Depth First Search (DFS)



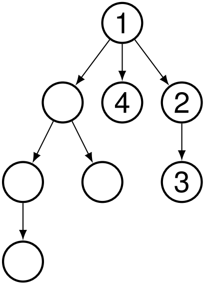
kiri ke kanan

kanan ke kiri

Recall: Depth First Search (DFS)

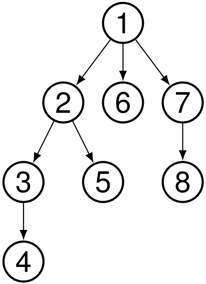


kiri ke kanan

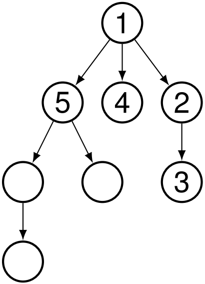


kanan ke kiri

Recall: Depth First Search (DFS)

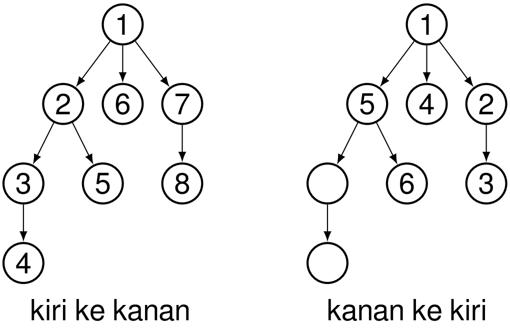


kiri ke kanan

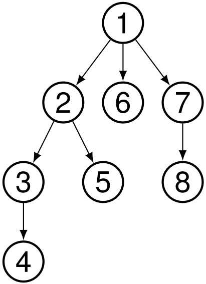


kanan ke kiri

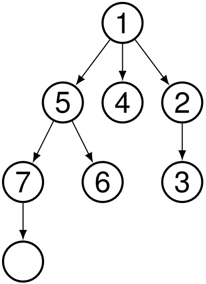
Recall: Depth First Search (DFS)



Recall: Depth First Search (DFS)

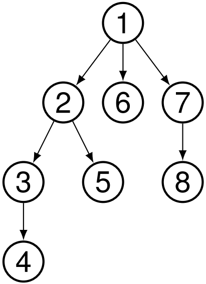


kiri ke kanan

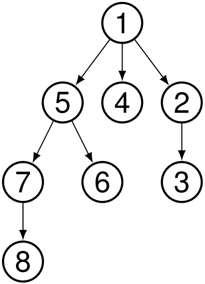


kanan ke kiri

Recall: Depth First Search (DFS)

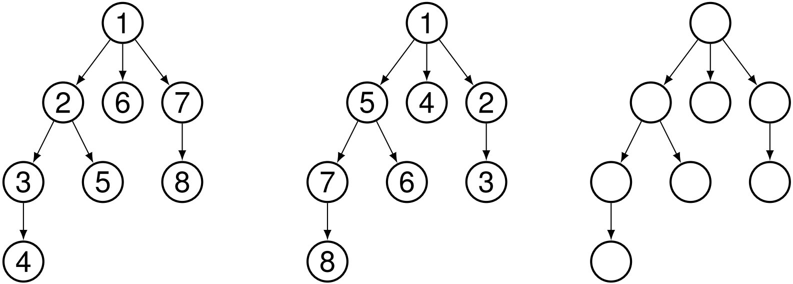


kiri ke kanan



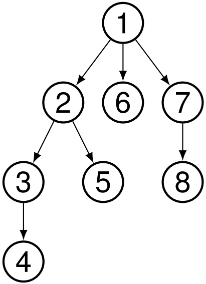
kanan ke kiri

Recall: Depth First Search (DFS)

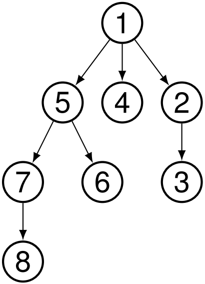


kiri ke kanan | kanan ke kiri | random

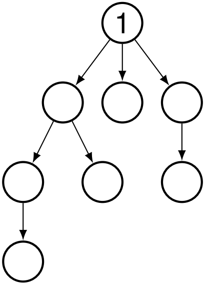
Recall: Depth First Search (DFS)



kiri ke kanan

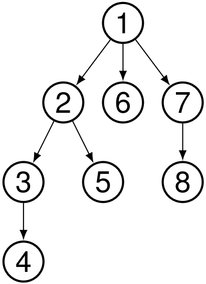


kanan ke kiri

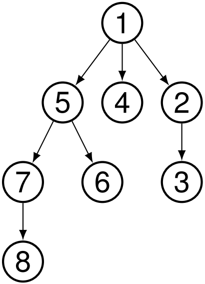


random

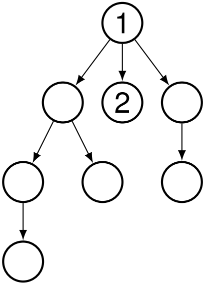
Recall: Depth First Search (DFS)



kiri ke kanan

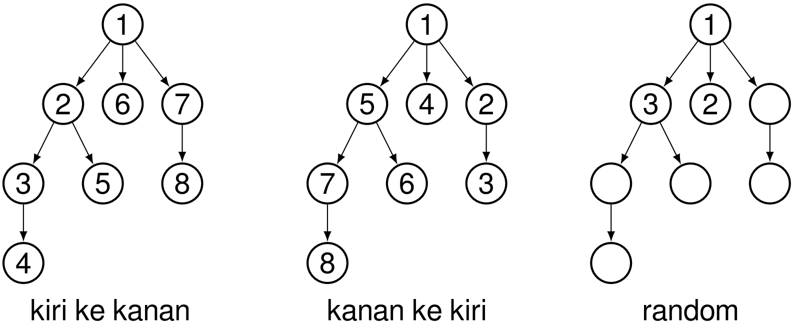


kanan ke kiri

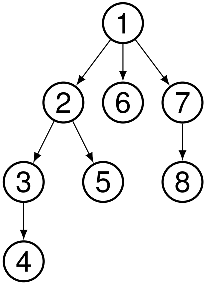


random

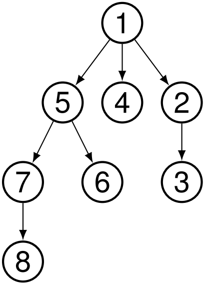
Recall: Depth First Search (DFS)



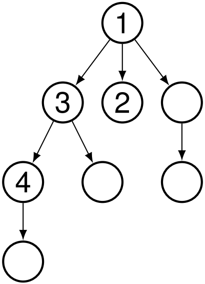
Recall: Depth First Search (DFS)



kiri ke kanan

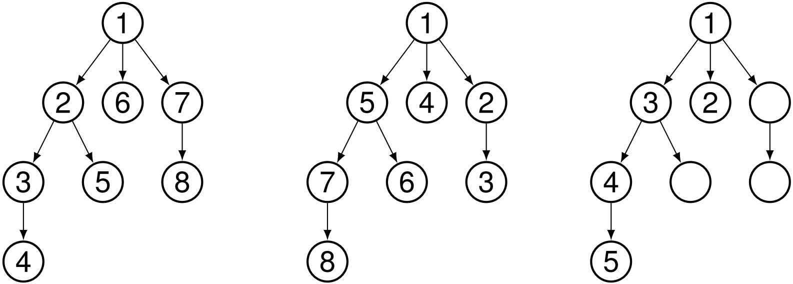


kanan ke kiri



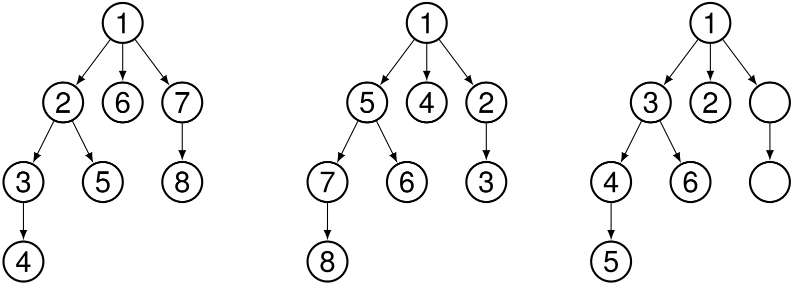
random

Recall: Depth First Search (DFS)



kiri ke kanan | kanan ke kiri | random

Recall: Depth First Search (DFS)

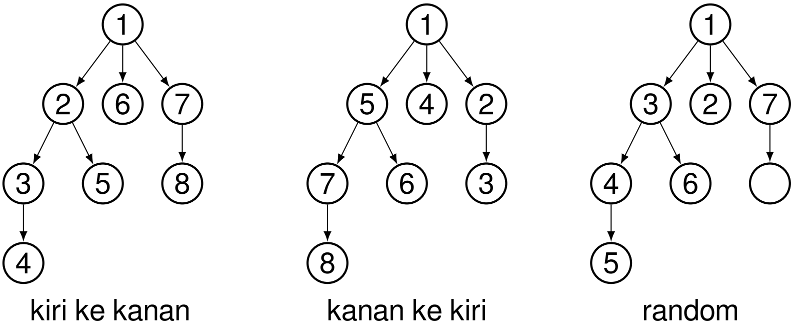


kiri ke kanan

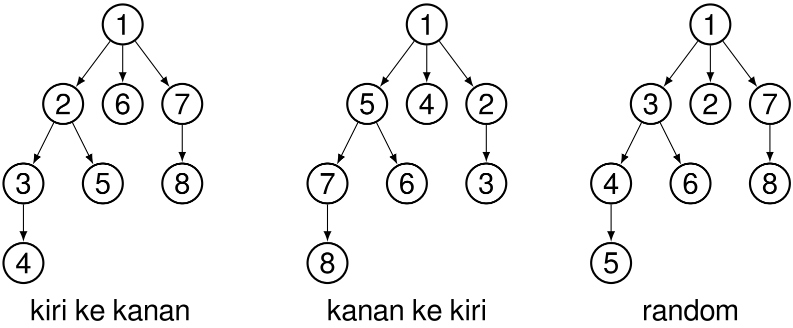
kanan ke kiri

random

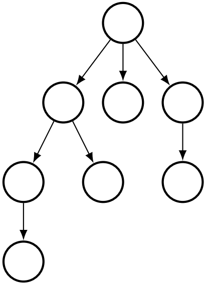
Recall: Depth First Search (DFS)



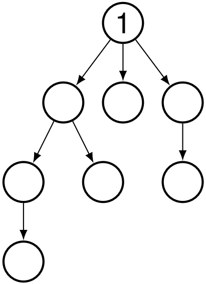
Recall: Depth First Search (DFS)



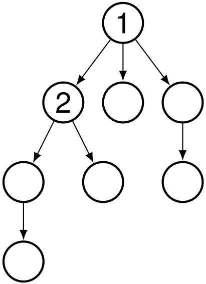
Recall: Breadth First Search (BFS)



Recall: Breadth First Search (BFS)

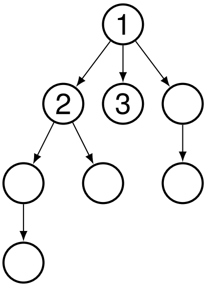


Recall: Breadth First Search (BFS)

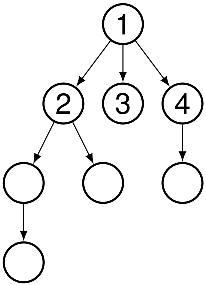


Fakultas Informatika
School of Computing
Telkom University

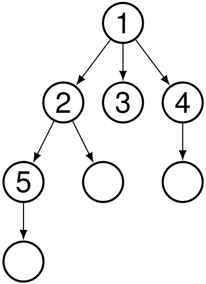
Recall: Breadth First Search (BFS)



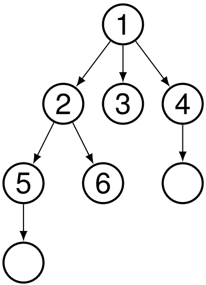
Recall: Breadth First Search (BFS)



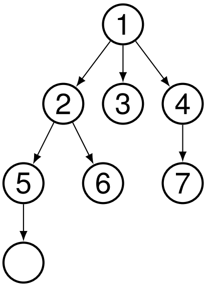
Recall: Breadth First Search (BFS)



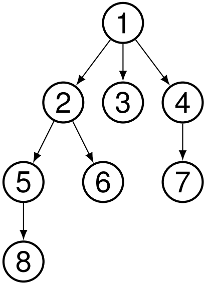
Recall: Breadth First Search (BFS)



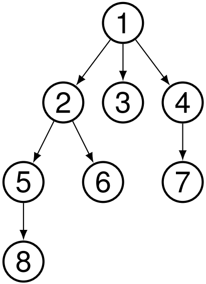
Recall: Breadth First Search (BFS)



Recall: Breadth First Search (BFS)

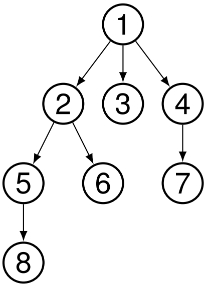


Recall: Breadth First Search (BFS)

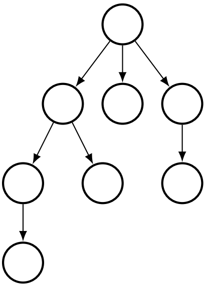


kiri ke kanan

Recall: Breadth First Search (BFS)

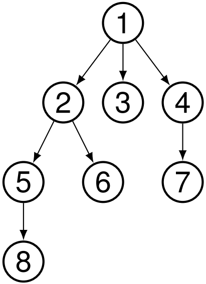


kiri ke kanan

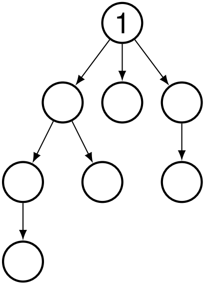


kanan ke kiri

Recall: Breadth First Search (BFS)

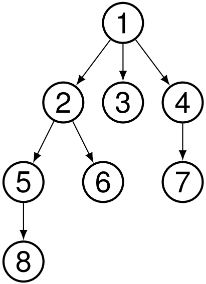


kiri ke kanan

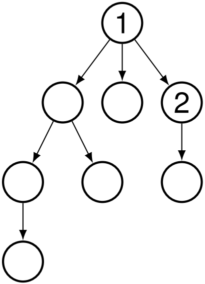


kanan ke kiri

Recall: Breadth First Search (BFS)

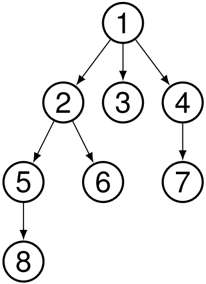


kiri ke kanan

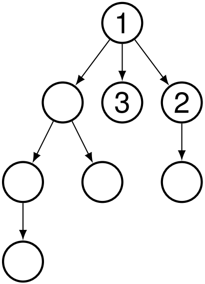


kanan ke kiri

Recall: Breadth First Search (BFS)

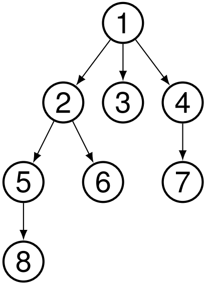


kiri ke kanan

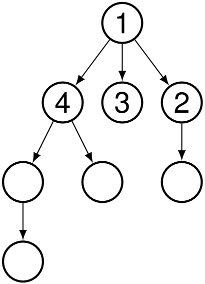


kanan ke kiri

Recall: Breadth First Search (BFS)

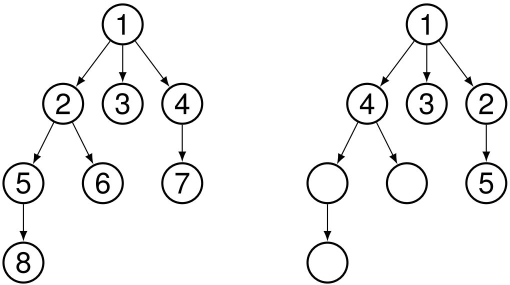


kiri ke kanan



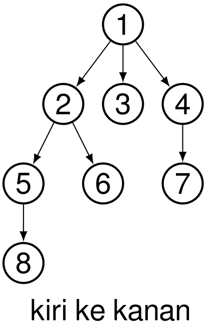
kanan ke kiri

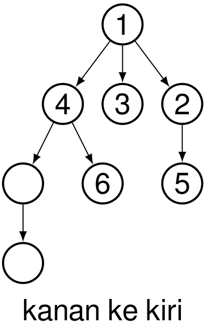
Recall: Breadth First Search (BFS)



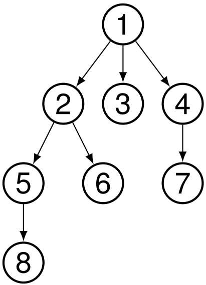
kiri ke kanan   kanan ke kiri

Recall: Breadth First Search (BFS)

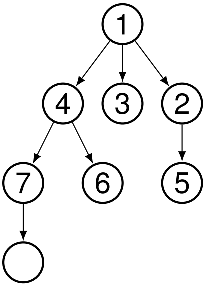




Recall: Breadth First Search (BFS)

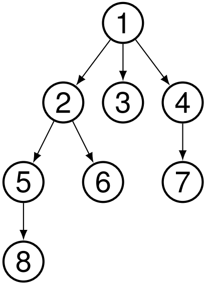


kiri ke kanan

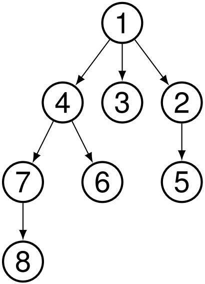


kanan ke kiri

Recall: Breadth First Search (BFS)

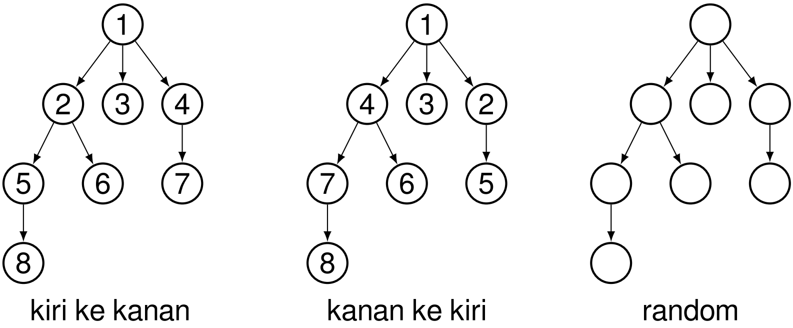


kiri ke kanan

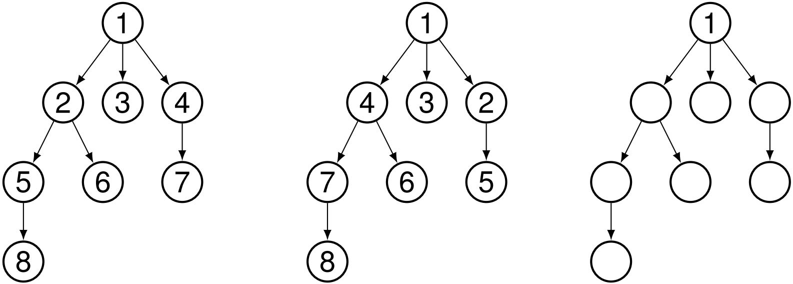


kanan ke kiri

Recall: Breadth First Search (BFS)

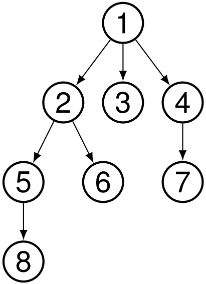


Recall: Breadth First Search (BFS)

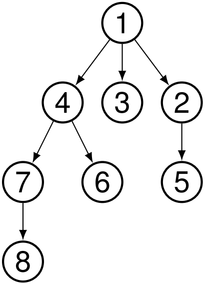


kiri ke kanan | kanan ke kiri | random

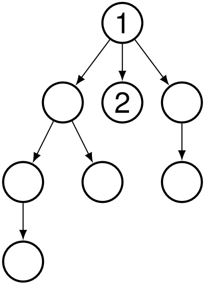
Recall: Breadth First Search (BFS)



kiri ke kanan

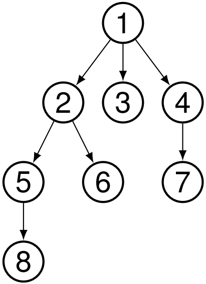


kanan ke kiri

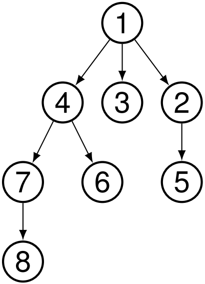


random

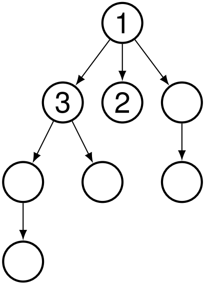
Recall: Breadth First Search (BFS)



kiri ke kanan

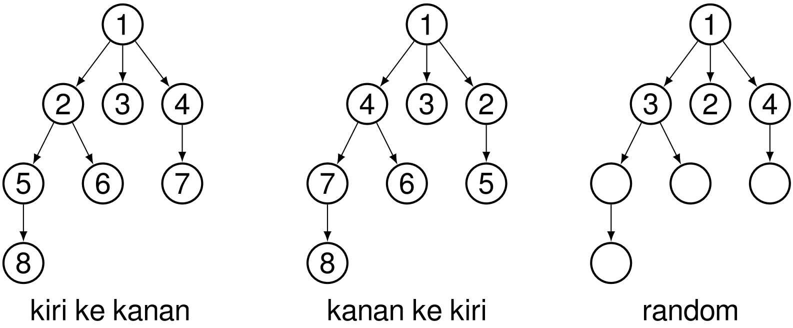


kanan ke kiri

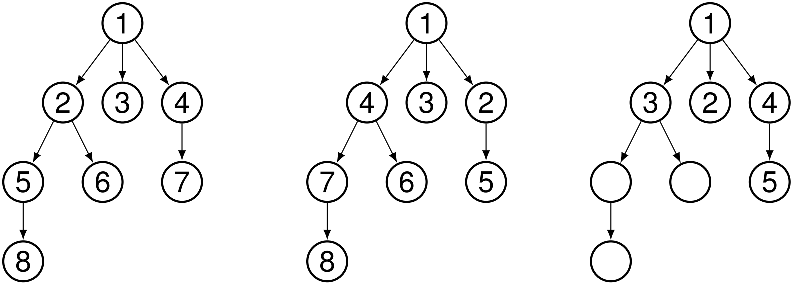


random

Recall: Breadth First Search (BFS)

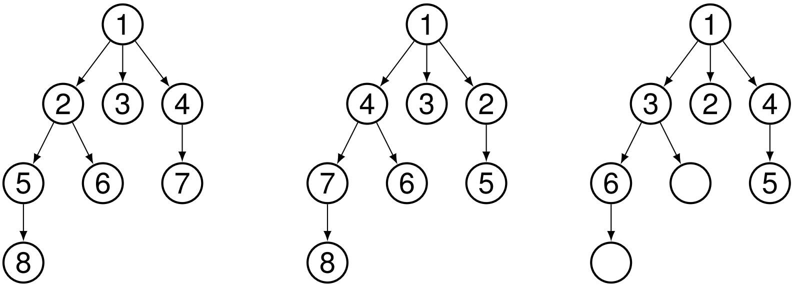


Recall: Breadth First Search (BFS)



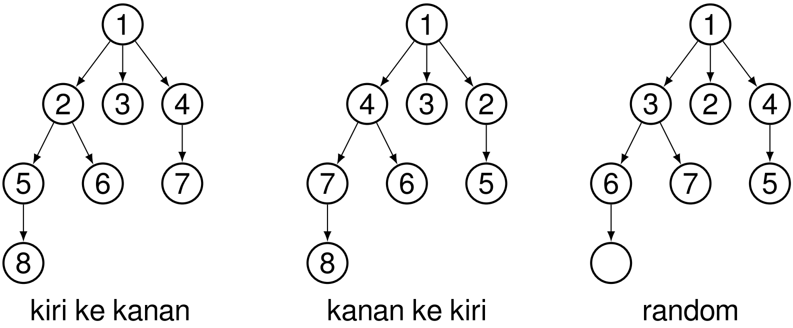
kiri ke kanan | kanan ke kiri | random

Recall: Breadth First Search (BFS)

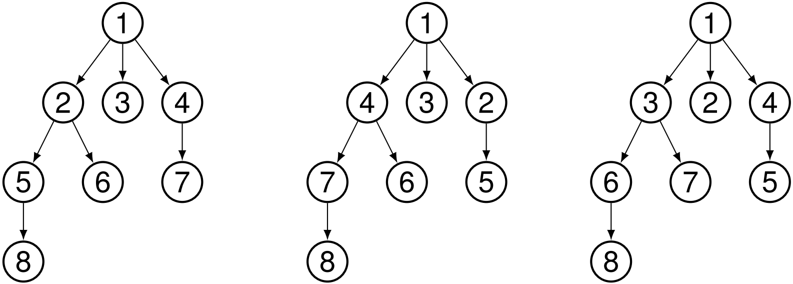


kiri ke kanan	kanan ke kiri	random

Recall: Breadth First Search (BFS)



Recall: Breadth First Search (BFS)



kiri ke kanan | kanan ke kiri | random

Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:

(*,*,*)

Contoh *state-space tree*

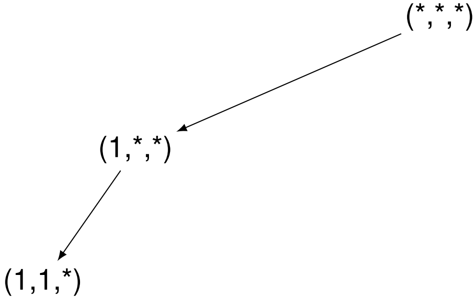
State space tree untuk permutasi tiga obyek:

(*,*,*)

(1,*,*)

Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:

(*,*,*)

(1,*,*)

~~(1,1,*)~~

Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:

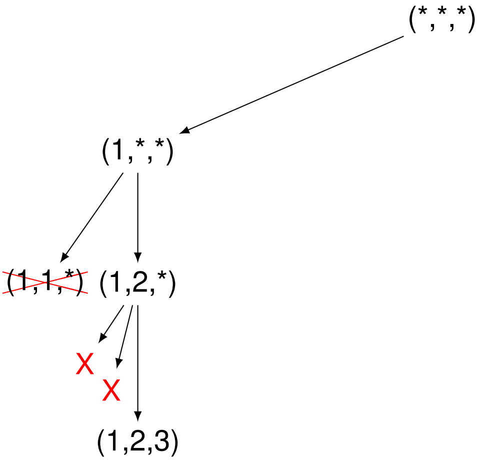
(*,*,*)

(1,*,*)

~~(1,1,*)~~ (1,2,*)

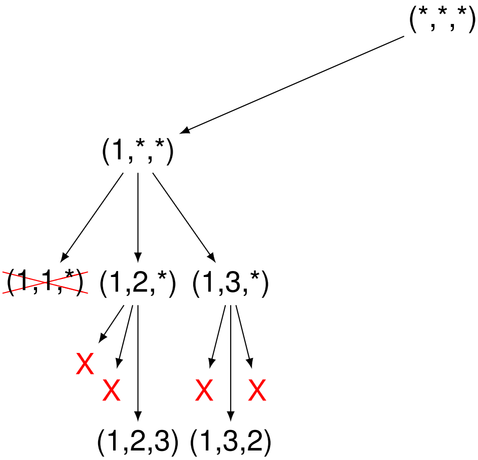
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



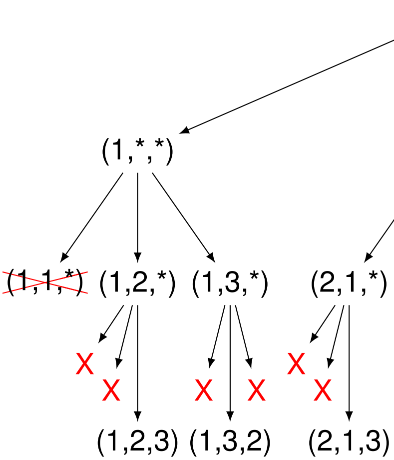
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



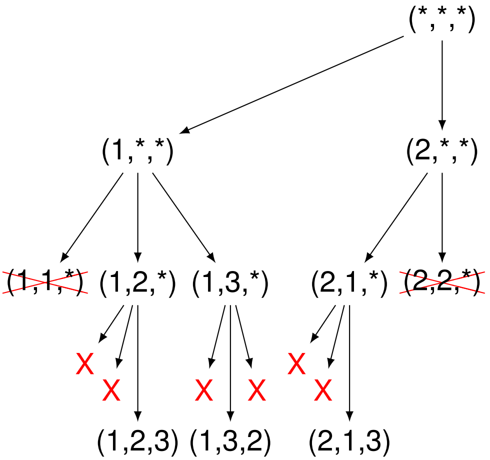
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



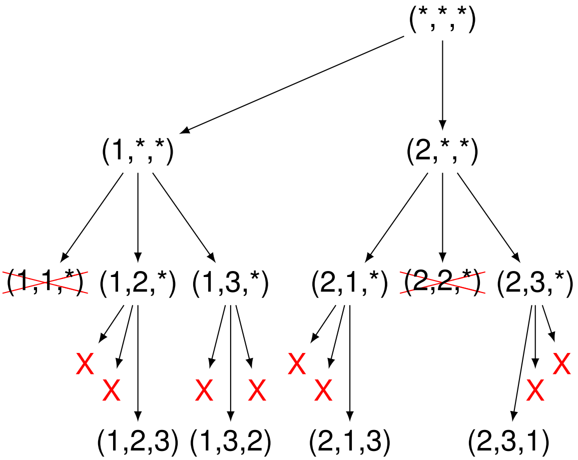
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



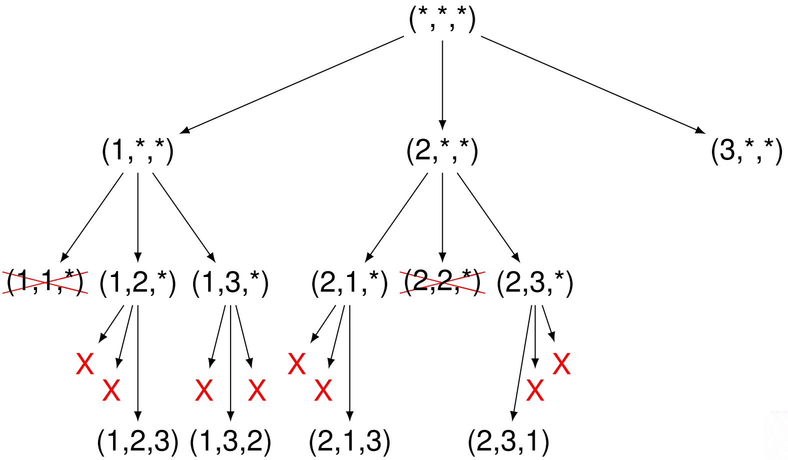
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



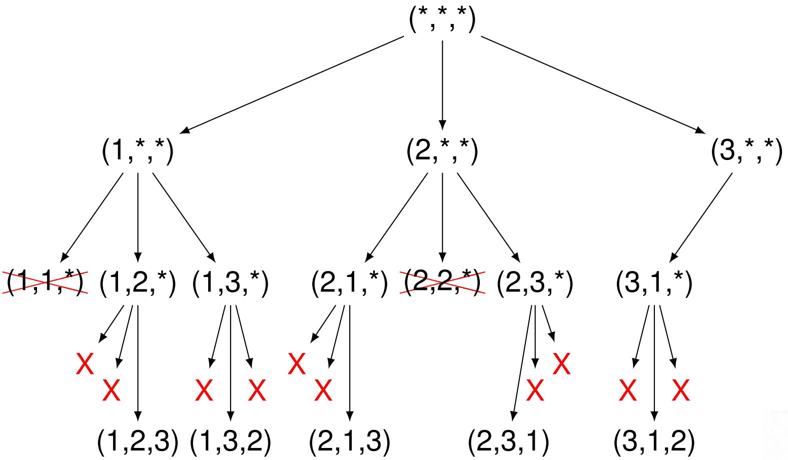
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



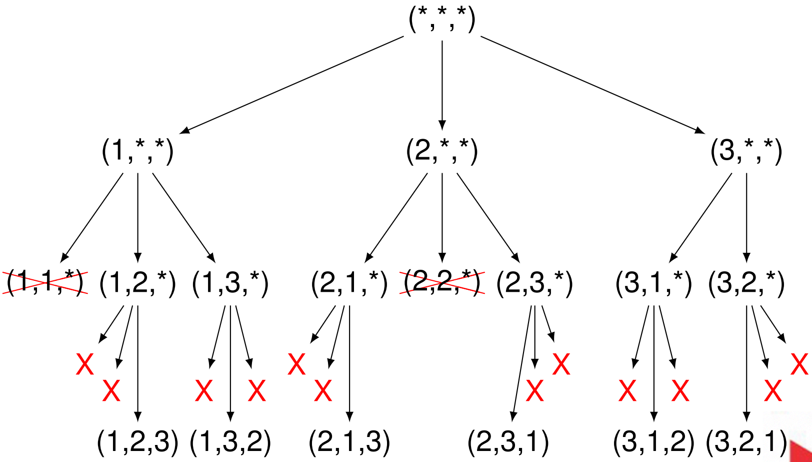
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



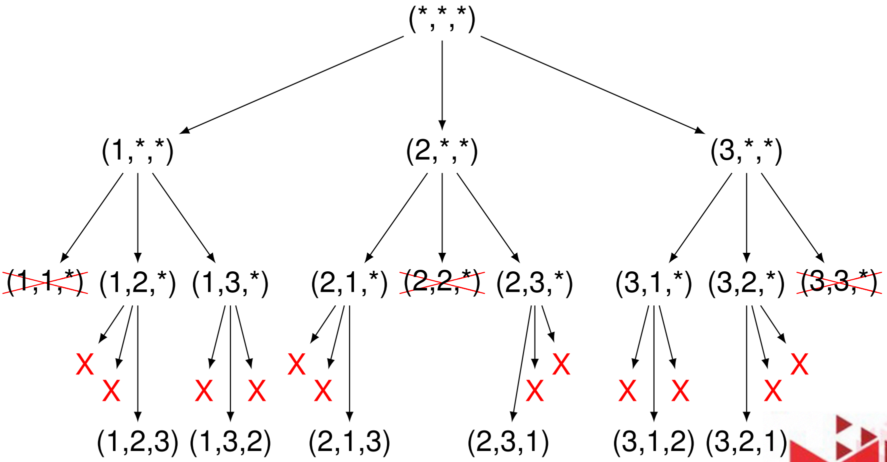
Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



Contoh *state-space tree*

State space tree untuk permutasi tiga obyek:



Permasalahan n-Queen

n-Queens Problem

Permasalahan:
Terdapat papan catur berukuran $n \times n$.
Bagaimana kita dapat menempatkan n buah ratu pada papan catur tersebut sedemikian sehingga tidak ada dua ratu yang dapat saling menyerang (i.e. tidak ada dua ratu yang terletak pada baris, kolom, ataupun diagonal yang sama).

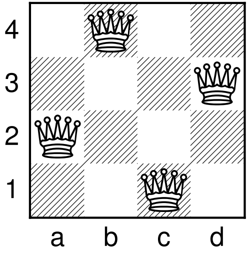
n-Queens Problem

Permasalahan:

Terdapat papan catur berukuran $n \times n$.

Bagaimana kita dapat menempatkan n buah ratu pada papan catur tersebut sedemikian sehingga tidak ada dua ratu yang dapat saling menyerang (i.e. tidak ada dua ratu yang terletak pada baris, kolom, ataupun diagonal yang sama).

Contoh solusi untuk $n = 4$:



Solusi dengan Brute Force (1)

Dengan Brute Force, kita dapat mengecek setiap kemungkinan penempatan n buah ratu di $n \times n$ petak yang tersedia, yaitu $C(n^2, n)$.

Untuk $n = 4$, terdapat $C(16, 4) = 1.820$ kemungkinan solusi

Untuk $n = 8$, terdapat $C(64, 8) = 4.426.165.368$ kemungkinan solusi

Solusi dengan Brute Force (2)

Alternatifnya, untuk setiap baris yang ada, kita dapat mengecek setiap kemungkinan penempatan 1 buah ratu di n petak yang tersedia (pada setiap barisnya).

Untuk $n = 4$, terdapat $(C(4,1))^4 = 4^4 = 256$ kemungkinan solusi

Untuk $n - 8$, terdapat $(C(8,1))^8 = 8^8 = 16.777.216$ kemungkinan solusi

Untuk n buah ratu, terdapat $(C(n,1))^n = n^n$ kemungkinan solusi

Solusi dengan Brute Force (3)

Dengan *exhaustive search*, kita dapat menyatakan solusinya dengan tupel

$$X = \{x_1, x_2, \ldots, x_n\}$$

dimana x_i menyatakan nomor kolom dimana ratu ditempatkan pada baris ke i

Untuk $n = 4$, terdapat $P(4,4) = 4! = 24$ kemungkinan solusi

Untuk $n - 8$, terdapat $P(8,4) = 8! = 40.320$ kemungkinan solusi

Untuk n buah ratu, terdapat $P(n,n) = n!$ kemungkinan solusi

Solusi dengan Backtracking

- Algoritma Backtracking memperbaiki algoritma brute force yang ke-3 (algoritma *exhaustive search*)
- Kita dapat menyatakan solusinya dengan tupel

$$X = \{x_1, x_2, \ldots, x_n\}$$

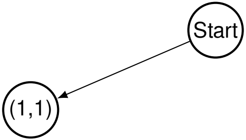
- Pada setiap tahap, kita bisa mengisi komponen pada tupel satu per satu
- Komponen yang dapat dipilih adalah $1, 2, \ldots, n$
- Maka, level pada space state tree menyatakan i (urutan pada permutasi), dimana pada setiap level, kita dapat memilih angka $1, 2, \ldots, n$ sebagai x_i

Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:

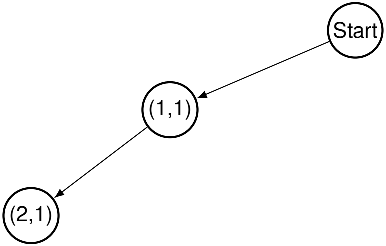
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



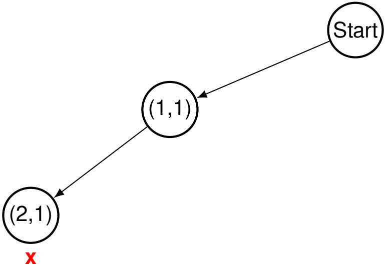
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



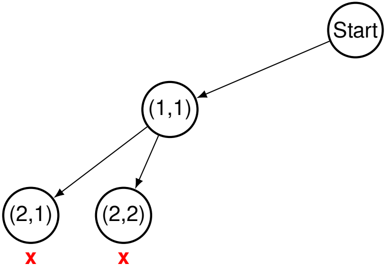
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



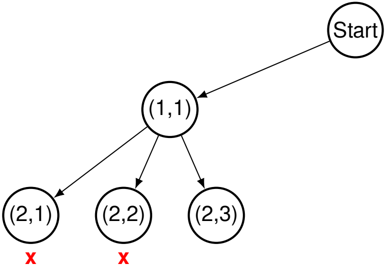
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



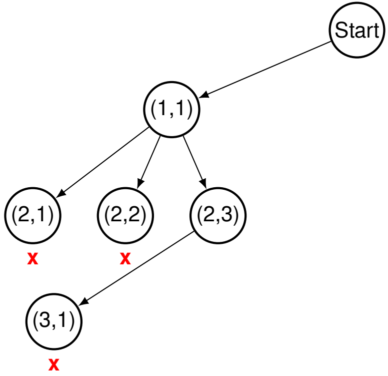
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

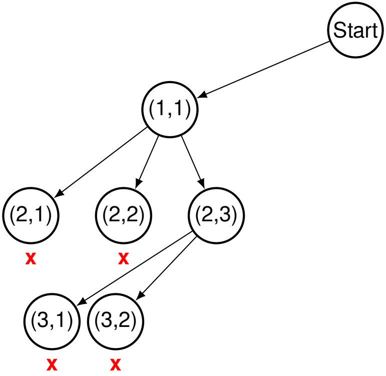
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

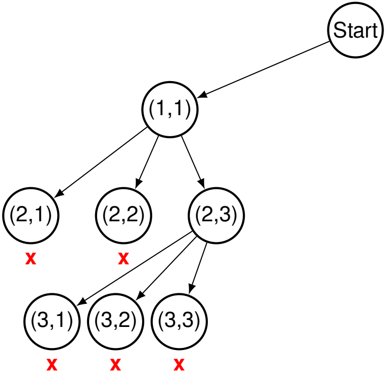
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

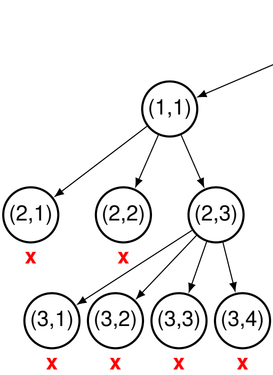
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

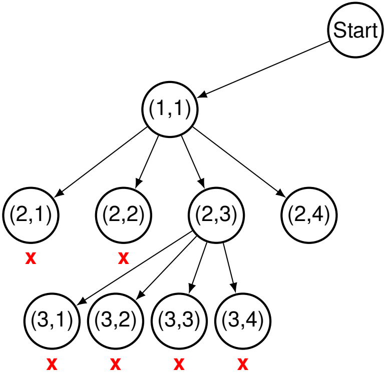
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

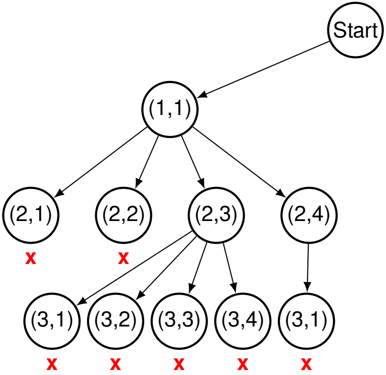
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

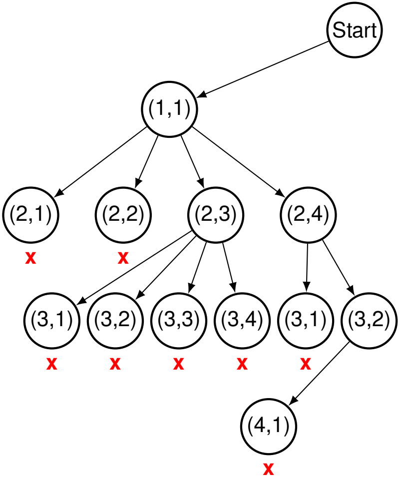
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

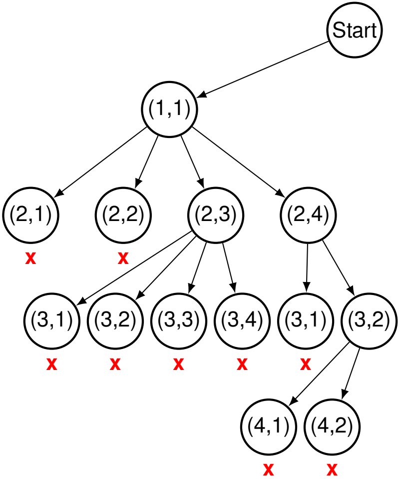
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

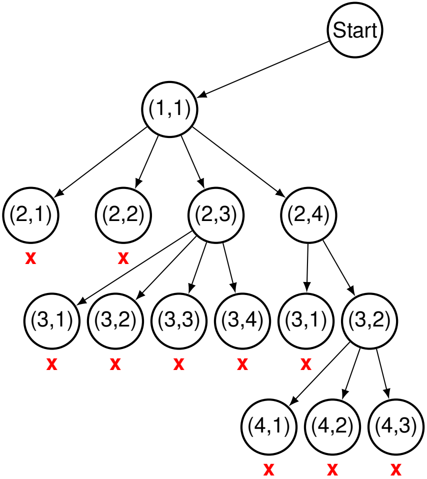
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

Maka didapatkan state-space tree sbb:



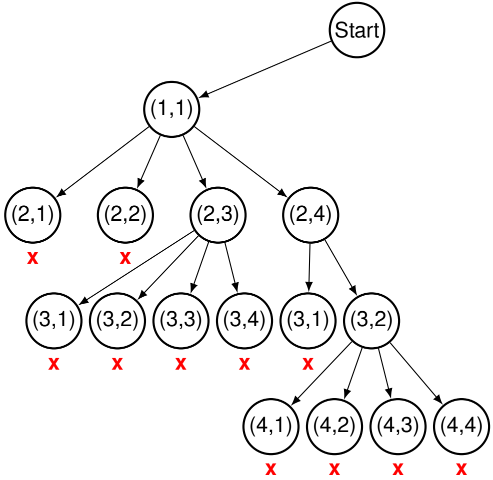
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

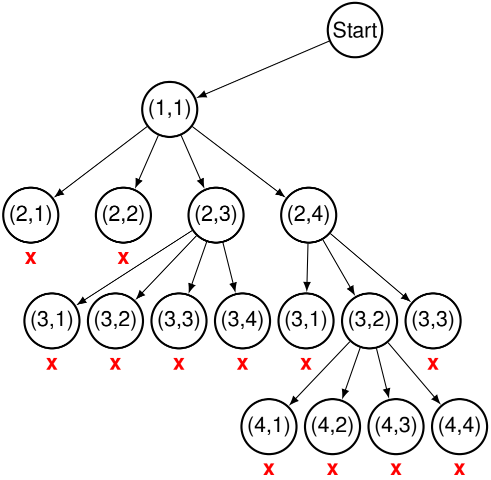
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

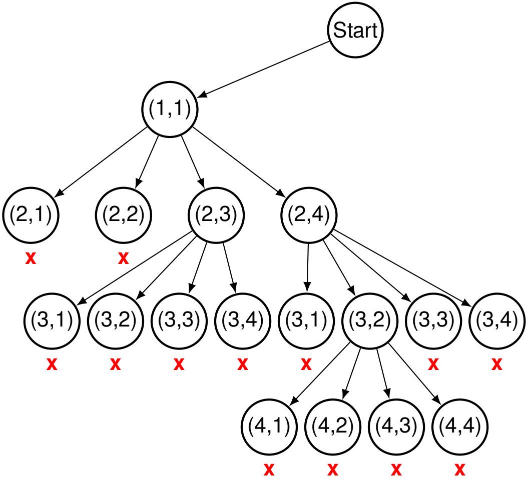
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

Maka didapatkan state-space tree sbb:



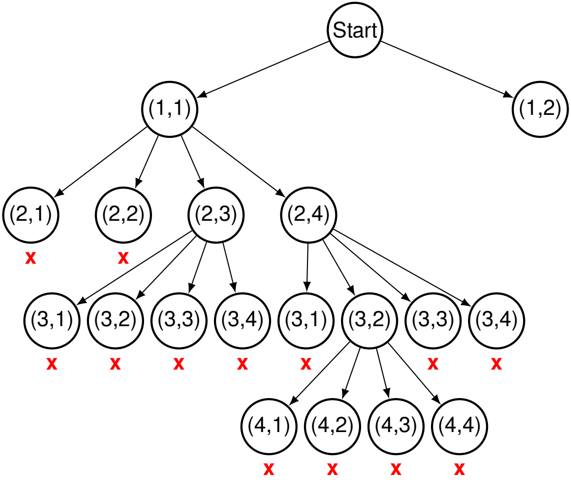
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



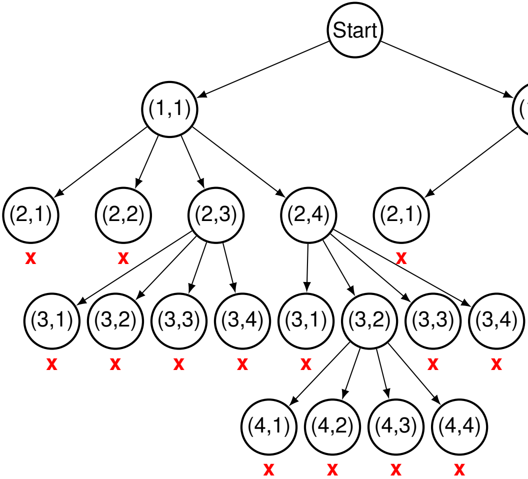
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



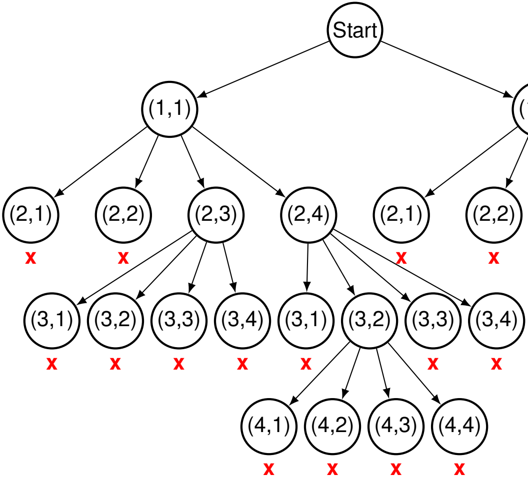
Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

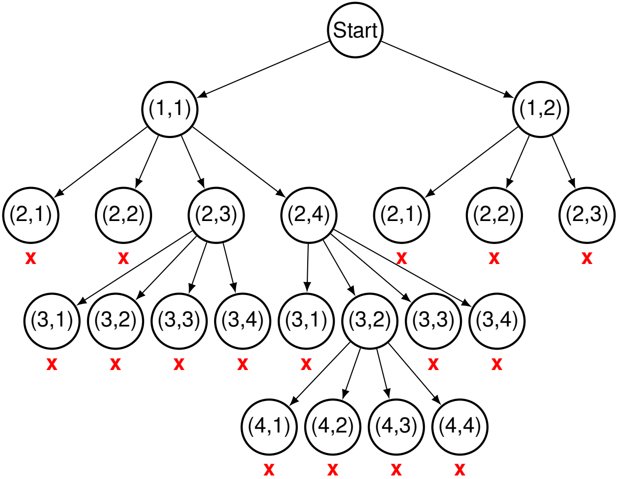
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

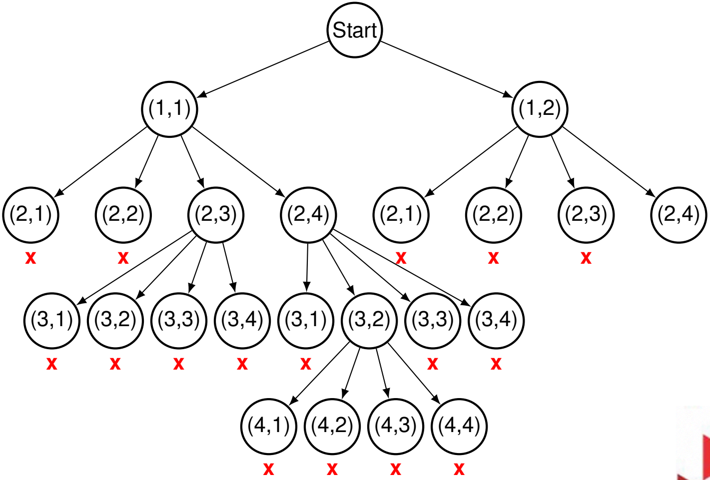
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

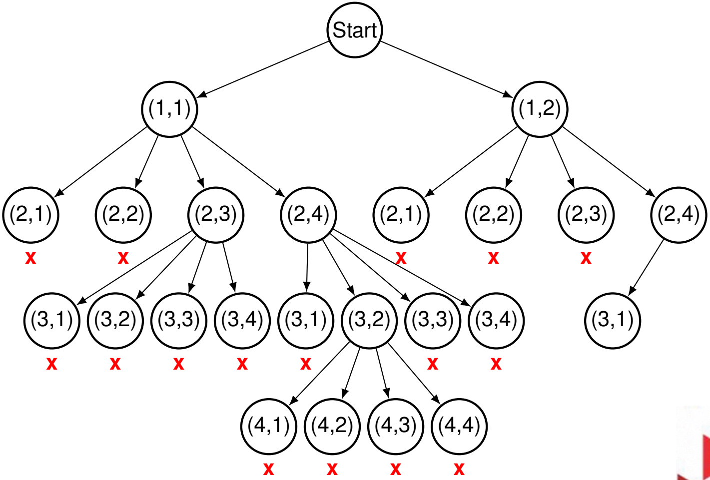
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

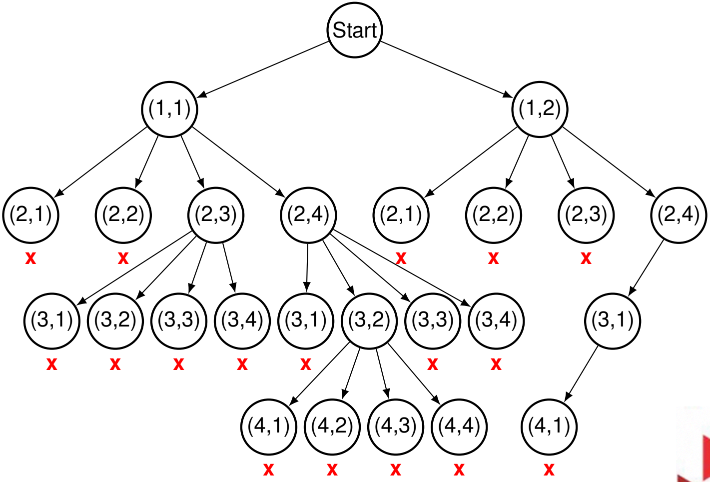
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

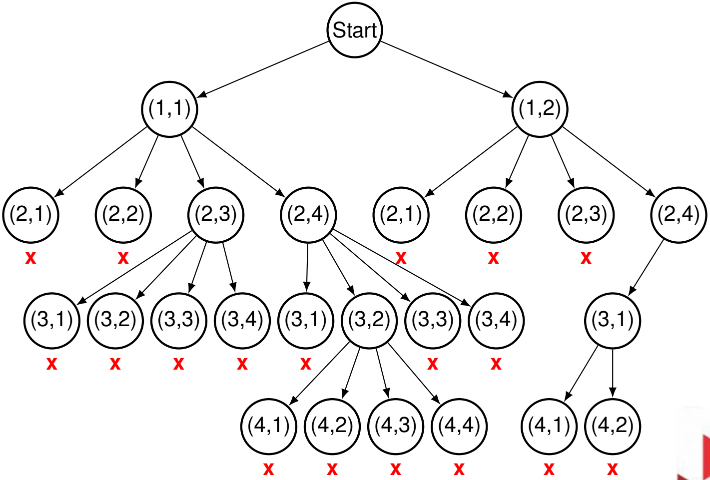
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

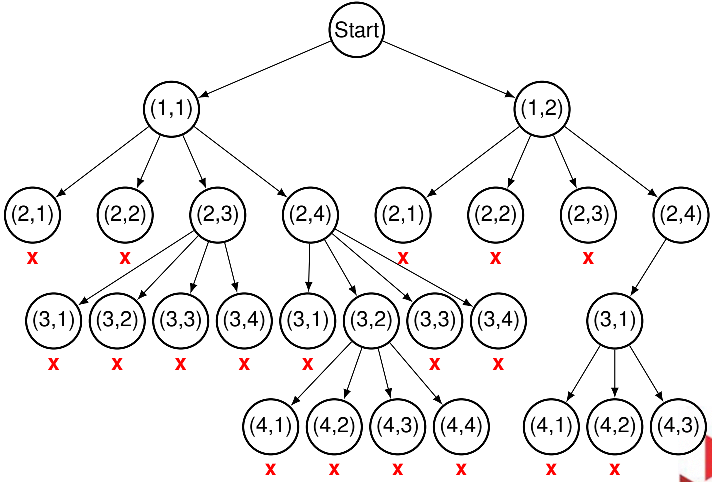
Maka didapatkan state-space tree sbb:



Contoh penyelesaian masalah 4-Queens (1)

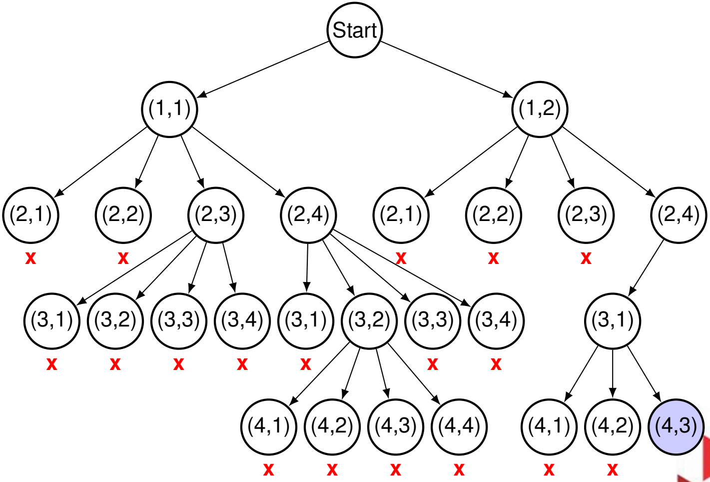
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.

Maka didapatkan state-space tree sbb:

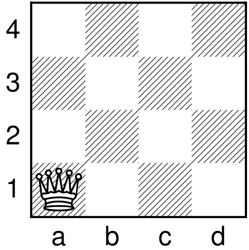


Contoh penyelesaian masalah 4-Queens (1)

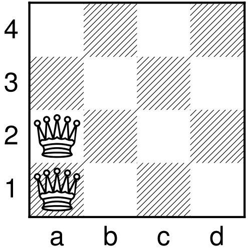
Misalkan simpul dilabeli dengan (i, x_i), yang menyatakan bahwa satu ratu ditempatkan pada baris ke-i dan kolom ke-x_i.
Maka didapatkan state-space tree sbb:



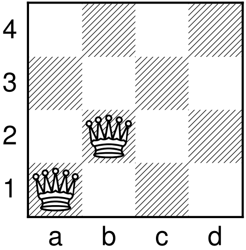
Contoh penyelesaian masalah 4-Queens (2)



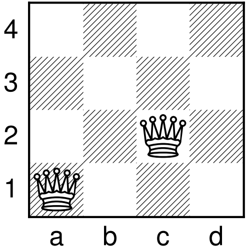
Contoh penyelesaian masalah 4-Queens (2)



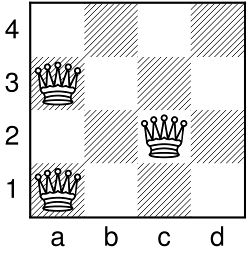
Contoh penyelesaian masalah 4-Queens (2)



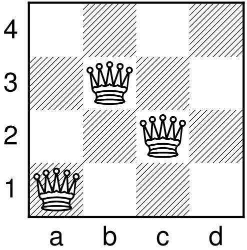
Contoh penyelesaian masalah 4-Queens (2)



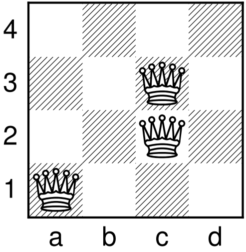
Contoh penyelesaian masalah 4-Queens (2)



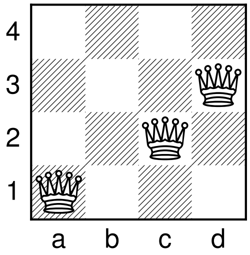
Contoh penyelesaian masalah 4-Queens (2)



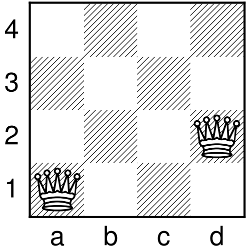
Contoh penyelesaian masalah 4-Queens (2)



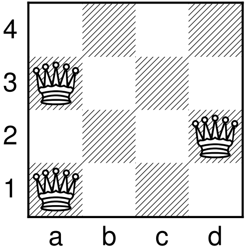
Contoh penyelesaian masalah 4-Queens (2)



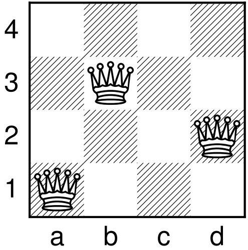
Contoh penyelesaian masalah 4-Queens (2)



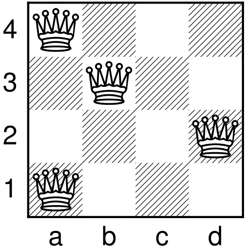
Contoh penyelesaian masalah 4-Queens (2)



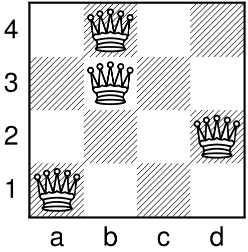
Contoh penyelesaian masalah 4-Queens (2)



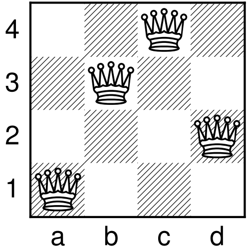
Contoh penyelesaian masalah 4-Queens (2)



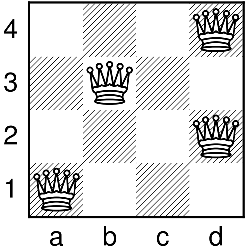
Contoh penyelesaian masalah 4-Queens (2)



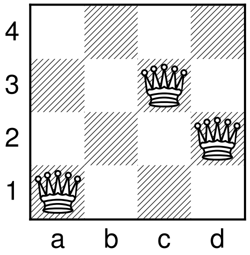
Contoh penyelesaian masalah 4-Queens (2)



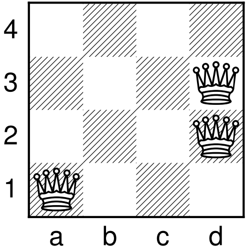
Contoh penyelesaian masalah 4-Queens (2)



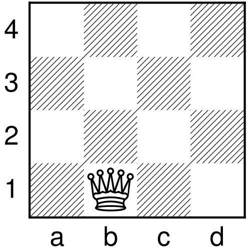
Contoh penyelesaian masalah 4-Queens (2)



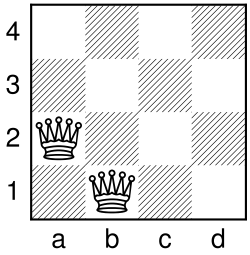
Contoh penyelesaian masalah 4-Queens (2)



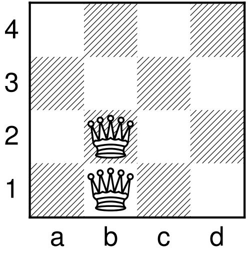
Contoh penyelesaian masalah 4-Queens (2)



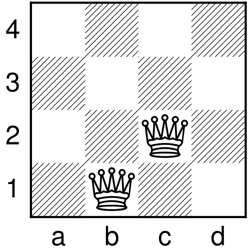
Contoh penyelesaian masalah 4-Queens (2)



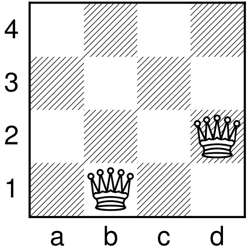
Contoh penyelesaian masalah 4-Queens (2)



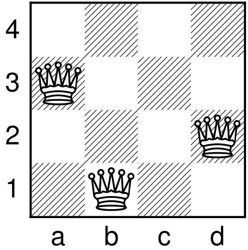
Contoh penyelesaian masalah 4-Queens (2)



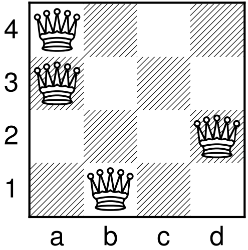
Contoh penyelesaian masalah 4-Queens (2)



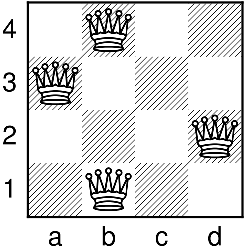
Contoh penyelesaian masalah 4-Queens (2)



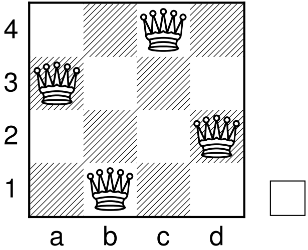
Contoh penyelesaian masalah 4-Queens (2)



Contoh penyelesaian masalah 4-Queens (2)



Contoh penyelesaian masalah 4-Queens (2)



Algoritma Backtracking permasalahan n-Queens

- Tinjau dua posisi ratu pada (i, j) dan (k, l)

Algoritma Backtracking permasalahan n-Queens

- Tinjau dua posisi ratu pada (i, j) dan (k, l)
- Jika dua ratu tersebut berada pada baris yang sama, maka $i = k$

Algoritma Backtracking permasalahan n-Queens

- Tinjau dua posisi ratu pada (i, j) dan (k, l)
- Jika dua ratu tersebut berada pada baris yang sama, maka $i = k$
- Jika dua ratu tersebut berada pada kolom yang sama, maka $j = l$

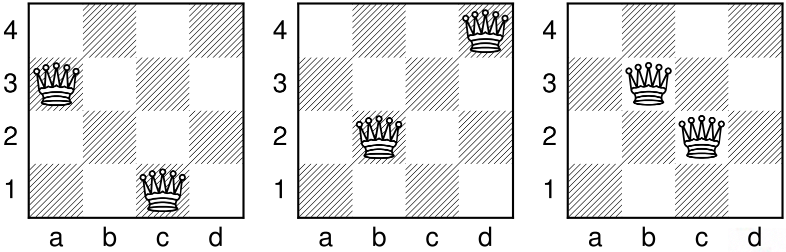
Algoritma Backtracking permasalahan n-Queens

- Tinjau dua posisi ratu pada (i, j) dan (k, l)
- Jika dua ratu tersebut berada pada baris yang sama, maka $i = k$
- Jika dua ratu tersebut berada pada kolom yang sama, maka $j = l$
- Jika dua ratu tersebut berada pada diagonal yang sama, maka $|i - k| = |j - l|$

Algoritma Backtracking permasalahan n-Queens

- Tinjau dua posisi ratu pada (i, j) dan (k, l)
- Jika dua ratu tersebut berada pada baris yang sama, maka $i = k$
- Jika dua ratu tersebut berada pada kolom yang sama, maka $j = l$
- Jika dua ratu tersebut berada pada diagonal yang sama, maka $|i - k| = |j - l|$

  Contoh:



Pseudo-code backtracking untuk n-Queens (1)

```
Procedure queens(i:index);
Var j:index;
Begin
    if promising(i) then
       if i=n then
          write(col[1] through col[n])
       else
          for j:=1 to n do
             col[i+1]:=j;
             queens(i+1)
          end
       end
    end
End;
```

Pseudo-code backtracking untuk n-Queens (2)

```
function promising(i:index):boolean;
Var k:index;
Begin
    k:=1;
    promising:=true;
    while k<i and promising do
       if col[i]=col[k] or abs(col[i]-col[k])=i-k then
          promising:=false
       end
       k:=k+1
    end
End;
```

Kompleksitas backtracking untuk n-Queens

Procedure `queens()` dapat dipanggil sebanyak banyaknya simpul pada *state-space tree*, yaitu:

- 1 simpul pada level 0
- n simpul pada level 1
- n^2 simpul pada level 2

$$\vdots$$

- n^n simpul pada level n

yaitu

$$\Sigma_{i=0}^{n} i^i = \frac{n^{n+1} - 1}{n - 1} \text{simpul}$$

Pencarian sirkuit Hamilton

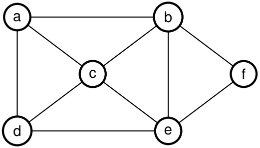
Pencarian Sirkuit Hamilton

Terdapat suatu graf G. Kita ingin mencari sirkuit Hamilton, yaitu sirkuit yang berawal dari satu simpul dan melewati setiap simpul lainnya pada G tepat satu kali sebelum kembali ke simpul asalnya.

Pencarian Sirkuit Hamilton

Terdapat suatu graf G. Kita ingin mencari sirkuit Hamilton, yaitu sirkuit yang berawal dari satu simpul dan melewati setiap simpul lainnya pada G tepat satu kali sebelum kembali ke simpul asalnya.

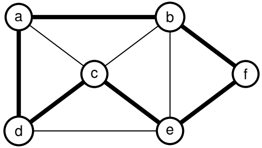
Contoh:



Pencarian Sirkuit Hamilton

Terdapat suatu graf G. Kita ingin mencari sirkuit Hamilton, yaitu sirkuit yang berawal dari satu simpul dan melewati setiap simpul lainnya pada G tepat satu kali sebelum kembali ke simpul asalnya.

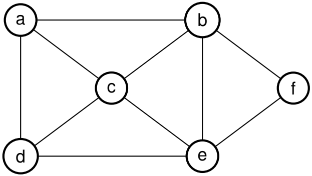
Contoh:



State-space tree untuk sirkuit Hamilton

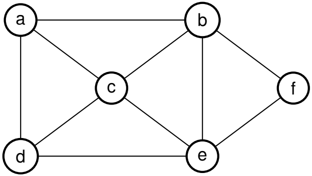
- Misalkan kita tinjau graf G dengan n simpul
- Sirkuit Hamilton dapat kita tulis sebagai $v_1 - v_2 - \ldots - v_n - v_1$, dimana $v_i \in V_G$
- Kita dapat mencari solusi dengan mengisi solusi per komponen untuk mendapatkan solusi parsial per tahapnya
- Artinya, pada setiap langkah ke-i, kita meninjau simpul ke-i yang perlu kita kunjungi pada sirkuit Hamilton
- Sehingga, level pohon menyatakan urutan kunjungan, dan (simpul) anak menyatakan simpul yang dapat dikunjungi pada tahap selanjutnya

Sirkuit Hamilton: contoh backtracking



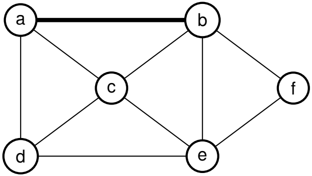
Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

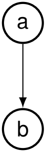
Sirkuit Hamilton: contoh backtracking



Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

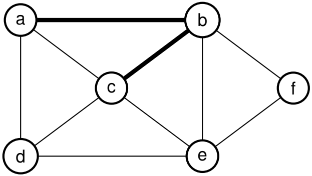
Sirkuit Hamilton: contoh backtracking

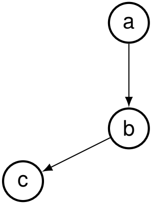




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

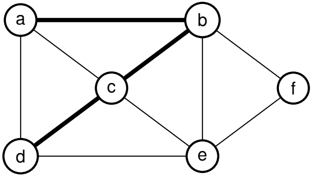
Sirkuit Hamilton: contoh backtracking

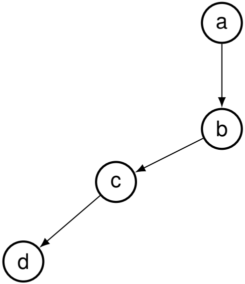




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

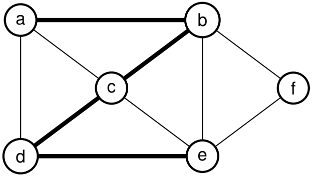
Sirkuit Hamilton: contoh backtracking

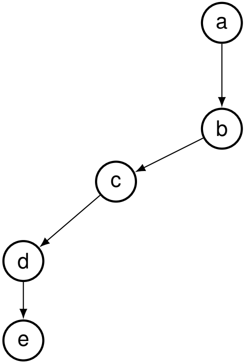




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

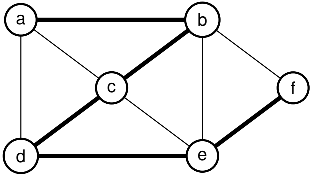
Sirkuit Hamilton: contoh backtracking

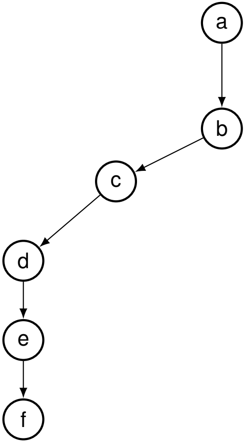




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

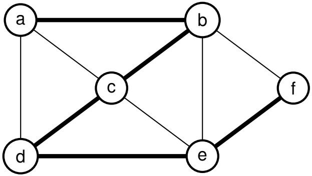
Sirkuit Hamilton: contoh backtracking

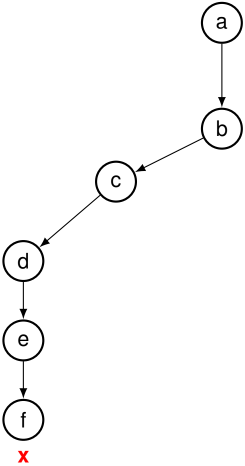




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

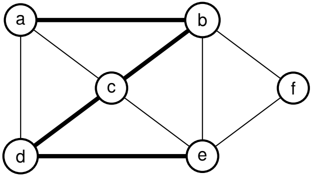
Sirkuit Hamilton: contoh backtracking

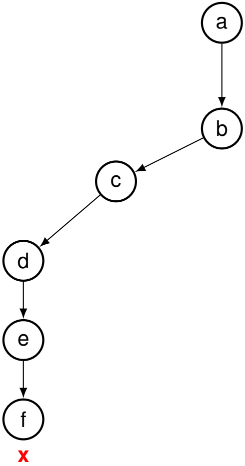




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

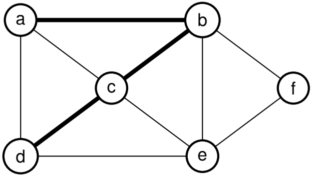
Sirkuit Hamilton: contoh backtracking

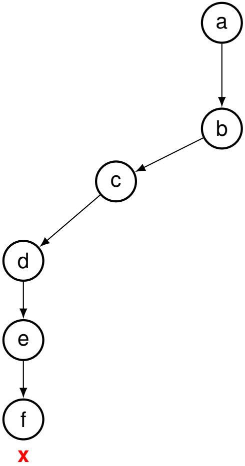




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

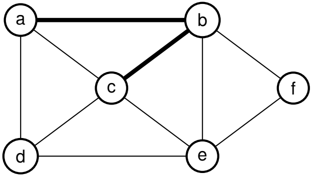
Sirkuit Hamilton: contoh backtracking

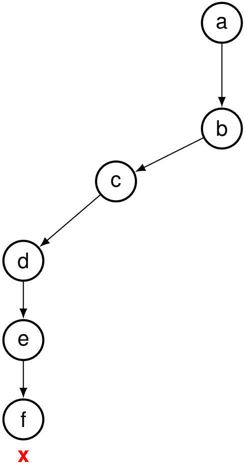




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

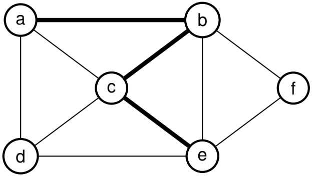
Sirkuit Hamilton: contoh backtracking

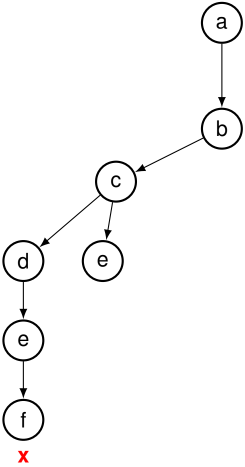




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

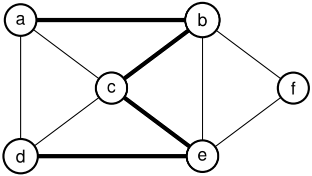
Sirkuit Hamilton: contoh backtracking

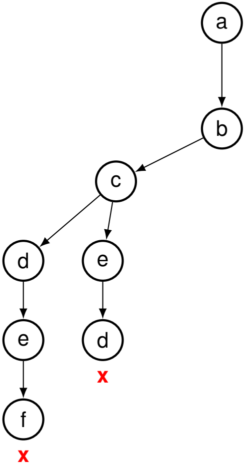




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

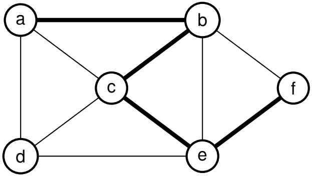
Sirkuit Hamilton: contoh backtracking

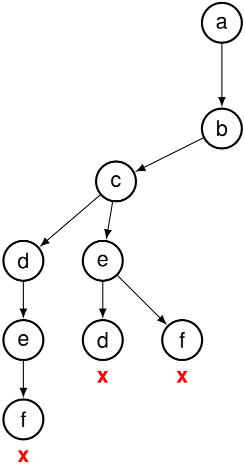




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

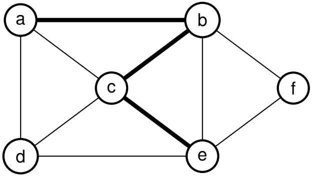
Sirkuit Hamilton: contoh backtracking

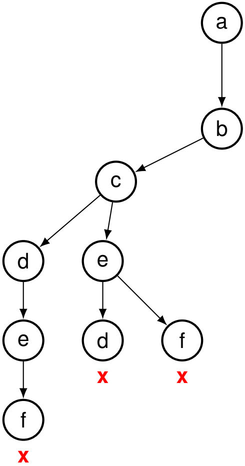




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

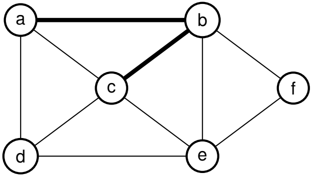
Sirkuit Hamilton: contoh backtracking

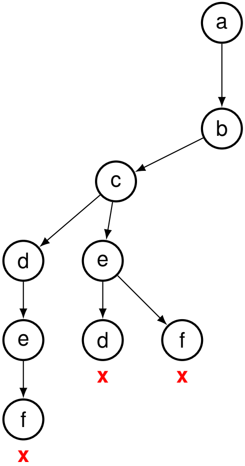




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

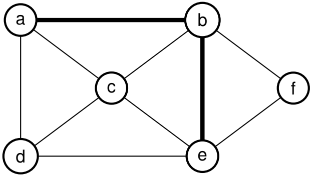
Sirkuit Hamilton: contoh backtracking

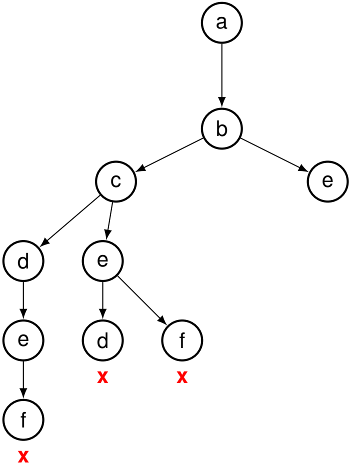




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

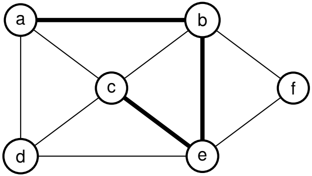
Sirkuit Hamilton: contoh backtracking

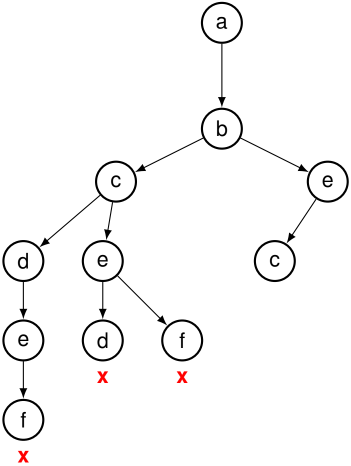




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

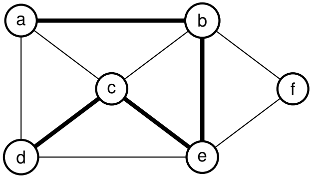
Sirkuit Hamilton: contoh backtracking

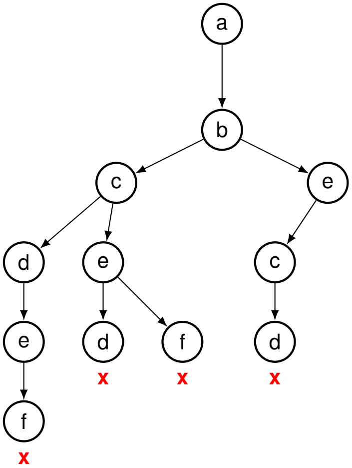




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

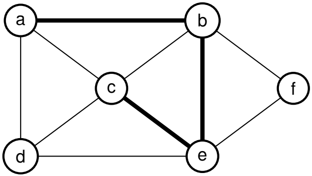
Sirkuit Hamilton: contoh backtracking

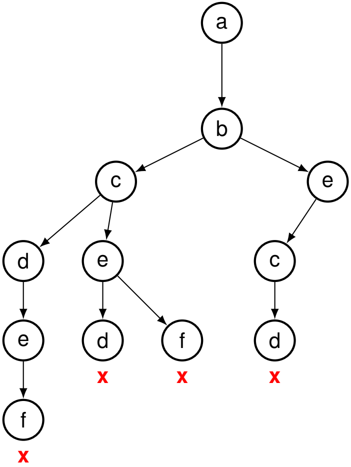




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

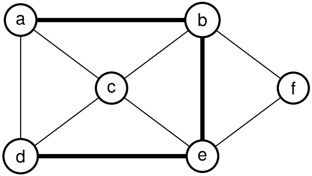
Sirkuit Hamilton: contoh backtracking

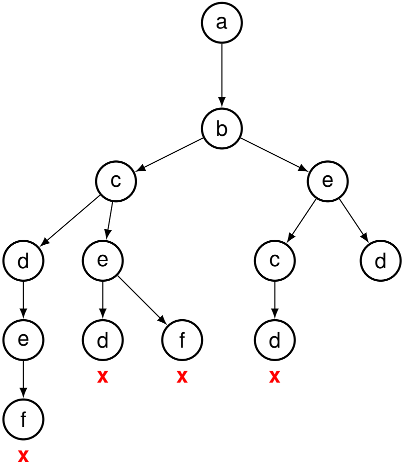




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

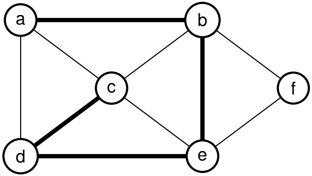
Sirkuit Hamilton: contoh backtracking

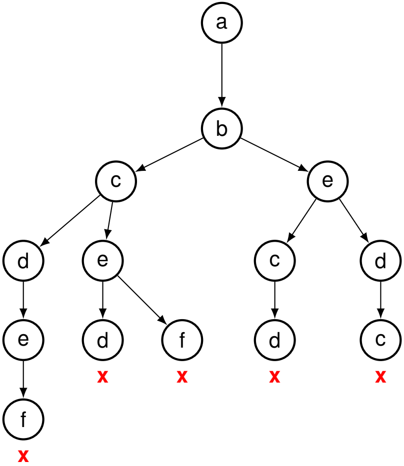




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

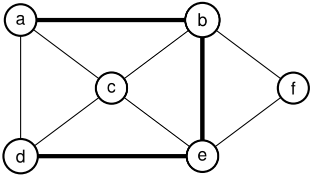
Sirkuit Hamilton: contoh backtracking

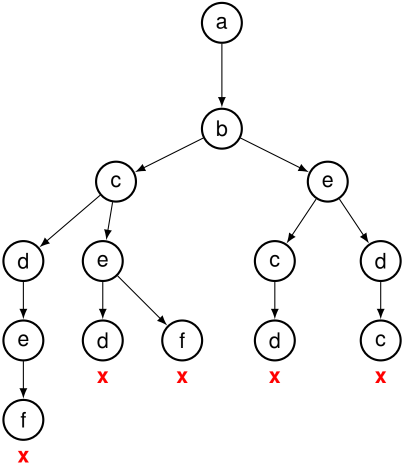




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

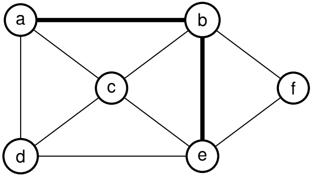
Sirkuit Hamilton: contoh backtracking

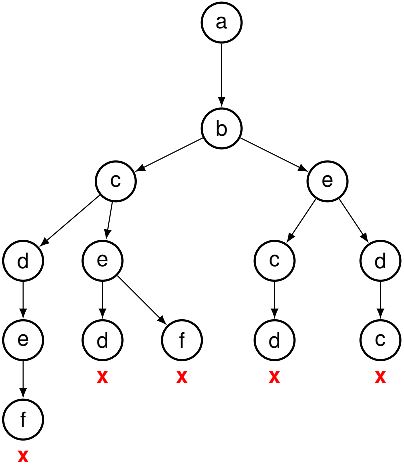




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

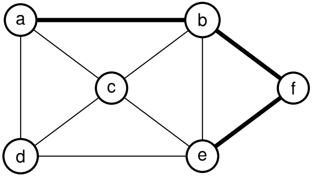
Sirkuit Hamilton: contoh backtracking

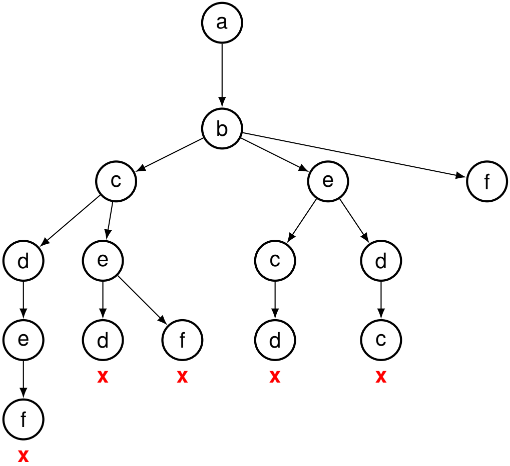




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

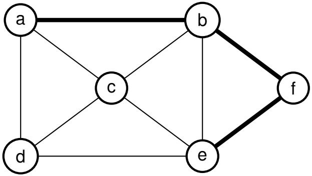
Sirkuit Hamilton: contoh backtracking

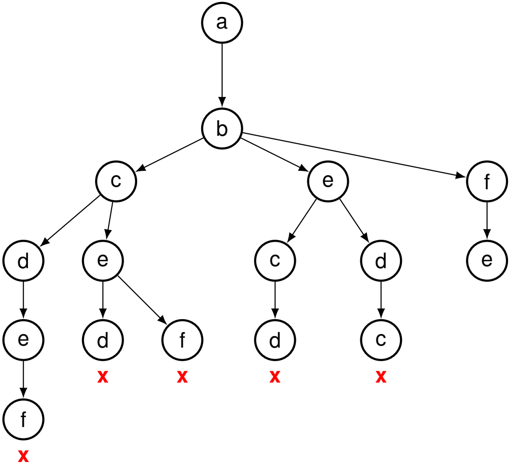




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

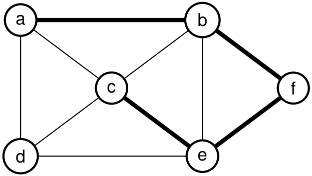
Sirkuit Hamilton: contoh backtracking

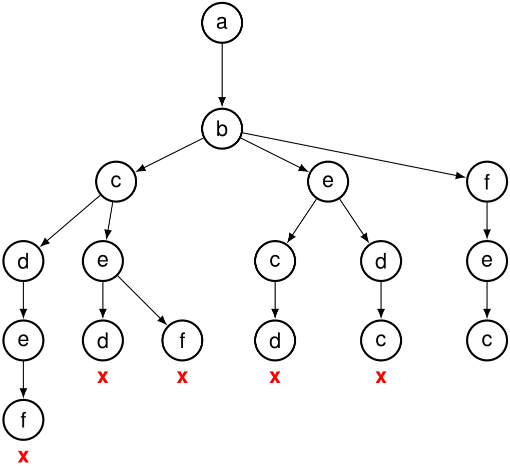




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

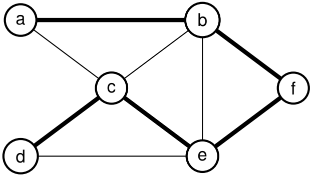
Sirkuit Hamilton: contoh backtracking

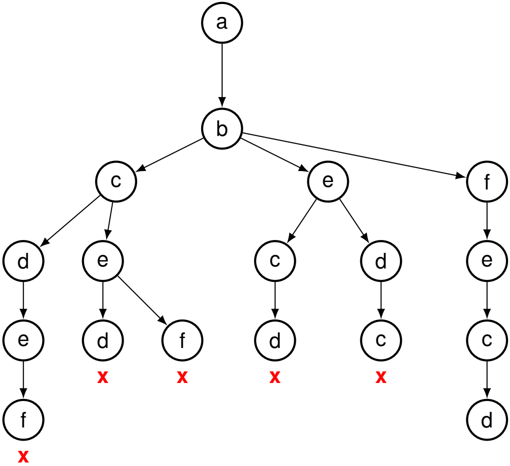




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

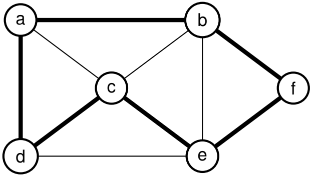
Sirkuit Hamilton: contoh backtracking

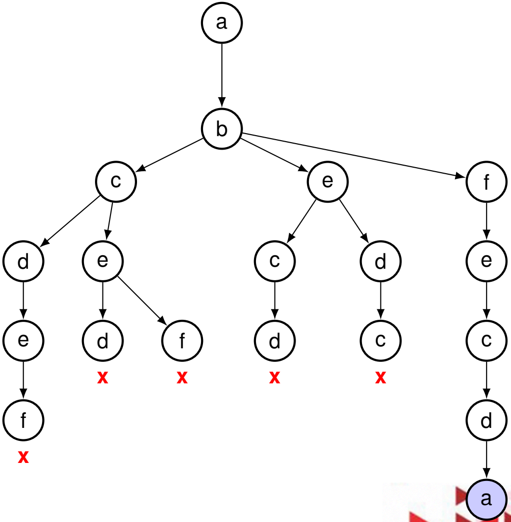




Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

Sirkuit Hamilton: contoh backtracking





Catatan: untuk menghemat tempat, pada setiap level, di sini kita hanya menulis simpul yang bertetangga dengan simpul yang dipilih sebelumnya namun belum pernah dipilih di solusi parsial yang sedang dievaluasi.

Sirkuit Hamilton: pseudo-code backtracking (1)

```
Procedure hamiltonian(i:index);
Var j:index;
Begin
    if promising(i) then
       if i=n-1 then
          write(vindex[1] through vindex[n-1])
       else
          for j:=2 to n do
             vindex[i+1]:=j;
             hamiltonian(i+1)
          end
       end
    end
End;
```

Sirkuit Hamilton: pseudo-code backtracking (2)

```
function promising(i:index):boolean;
Var k:index; promising: boolean
Begin
    if i=n-1 and no edge (vindex[i],vindex[0]) then
       promising:=false;
    else if i>0 and no edge (vindex[i-1],vindex[i]) then
       promising:=false;
    else
       k:=1;
       promising:=true;
       while k<i and promising do
          if vindex[i]=vindex[k] and promising then
             promising:=false
          end
          k:=k+1
       end
    end
End;
```

Kompleksitas backtracking untuk sirkuit Hamilton

Procedure `hamiltonian()` dapat dipanggil sebanyak banyaknya simpul pada ruang solusi, yaitu:

- 1 simpul pada level 0
- $n-1$ simpul pada level 1
- $(n-1)^2$ simpul pada level 2

$$\vdots$$

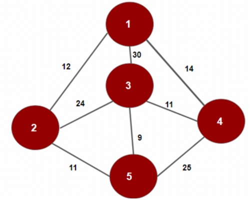
- $(n-1)^{n-1}$ simpul pada level $n-1$

yaitu

$$1 + (n-1) + (n-1)^2 + \ldots + (n-1)^{n-1} = \frac{(n-1)^n - 1}{n-2} \text{simpul}$$

Latihan (1)

Perhatikan graf berikut:



Dengan menggunakan teknik backtracking, carilah semua Sirkuit Hamiltonian yang ada pada graf tersebut, kemudian tentukan sirkuit mana yang memiliki total bobot terkecil.

Pewarnaan Graf

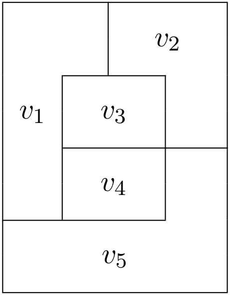
Permasalahan m-Colouring

Misalkan terdapat graf G dengan simpul $v_1, v_2, \ldots, v_n$. Misalkan terdapat m warna untuk mewarnai simpul-simpul tersebut. Bagaimana caranya kita dapat mewarnai simpul-simpul tersebut sedemikian sehingga tidak ada dua simpul bertetangga memiliki warna yang sama?

Catatan: tidak semua warna harus dipakai.

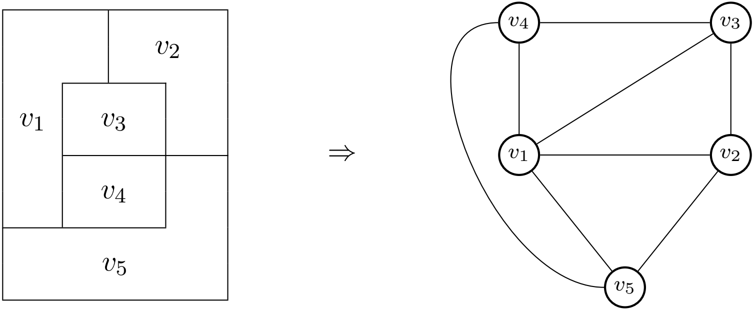
Contoh aplikasi pewarnaan graf

Contoh: pewarnaan peta



Contoh aplikasi pewarnaan graf

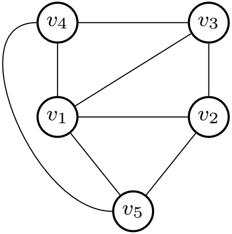
Contoh: pewarnaan peta



State-space tree untuk pewarnaan graf

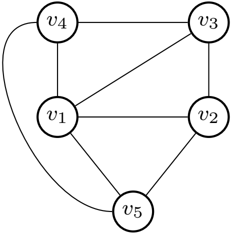
- Misalkan kita tinjau graf G dengan n simpul untuk pewarnaan graf dengan m warna (m-*colouring problem*)
- Solusi permasalahan pewarnaan graf dapat kita tulis sebagai tupel $\{x_1, x_2, \ldots, x_n\}$ dimana x_i menyatakan warna yang digunakan untuk mewarnai simpul ke-i
- Kita dapat mencari solusi dengan mengisi solusi per komponen untuk mendapatkan solusi parsial per tahapnya
- Artinya, pada setiap langkah ke$-i$, kita tinjau warna yang dapat digunakan untuk mewarnai simpul ke$-i$
- Sehingga, level pohon menyatakan simpul yang kita tinjau

Pewarnaan graf: contoh backtracking

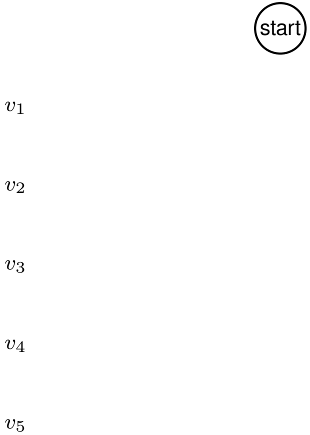


warna tersedia:
merah (1), biru (2), hijau (3)
($m = 3$)

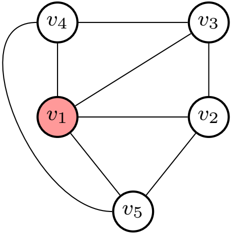
Pewarnaan graf: contoh backtracking



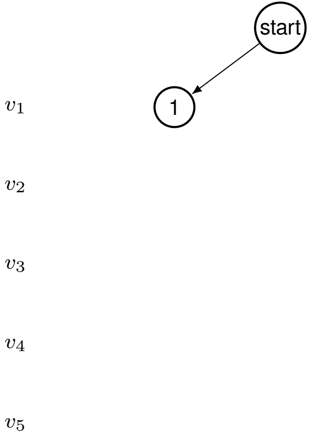
warna tersedia:
merah (1), biru (2), hijau (3)
($m = 3$)



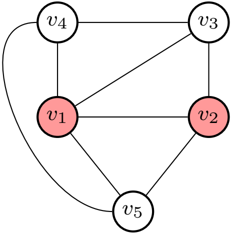
Pewarnaan graf: contoh backtracking



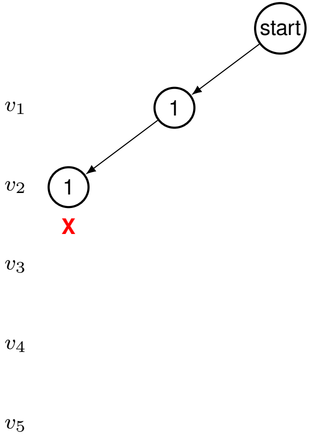
warna tersedia:
merah (1), biru (2), hijau (3)
($m = 3$)



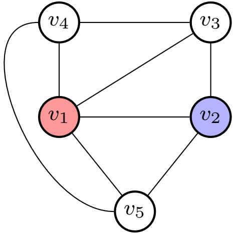
Pewarnaan graf: contoh backtracking



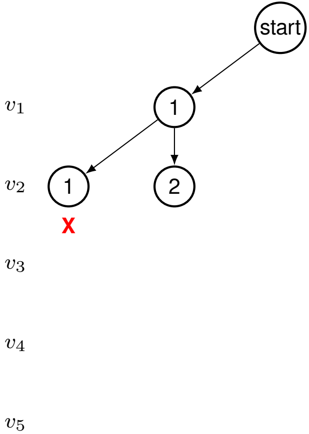
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



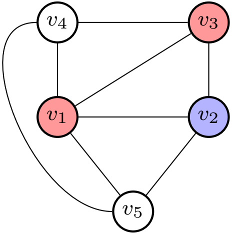
Pewarnaan graf: contoh backtracking



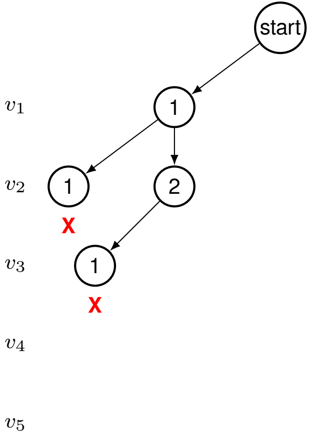
warna tersedia:
merah (1), biru (2), hijau (3)
($m = 3$)



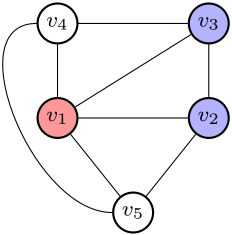
Pewarnaan graf: contoh backtracking



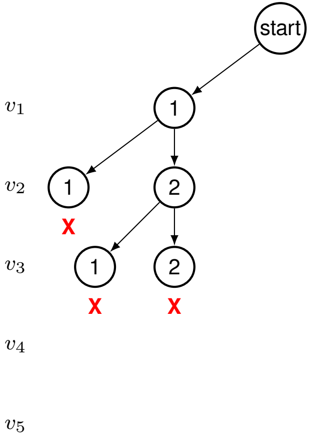
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



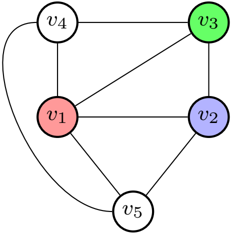
Pewarnaan graf: contoh backtracking



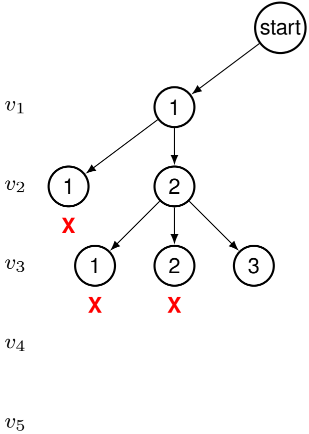
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



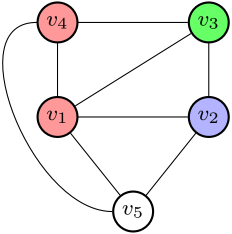
Pewarnaan graf: contoh backtracking



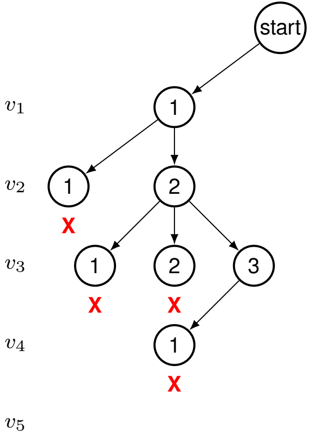
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



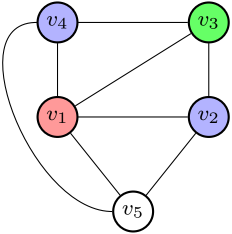
Pewarnaan graf: contoh backtracking



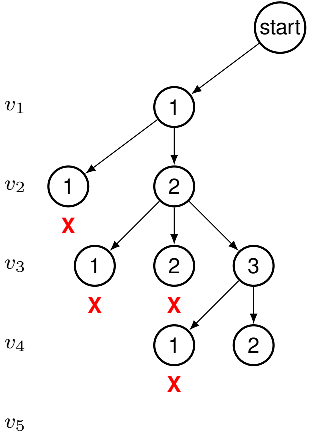
warna tersedia:
merah (1), biru (2), hijau (3)
($m = 3$)



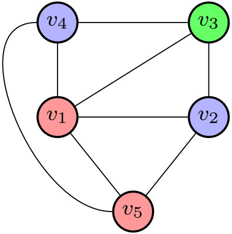
Pewarnaan graf: contoh backtracking



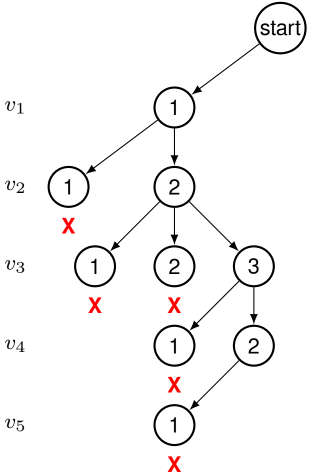
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



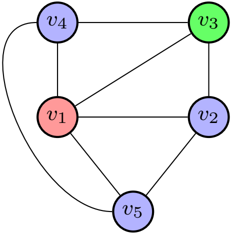
Pewarnaan graf: contoh backtracking



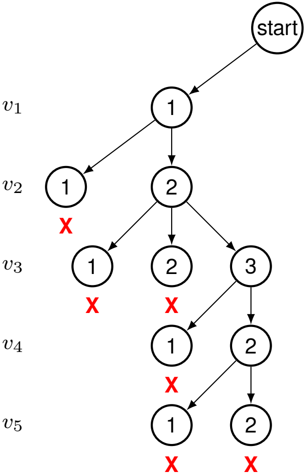
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



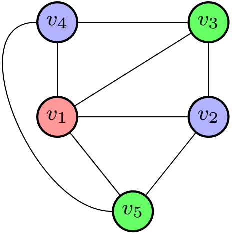
Pewarnaan graf: contoh backtracking



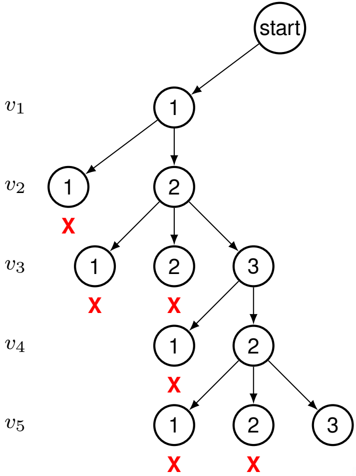
warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



Pewarnaan graf: contoh backtracking



warna tersedia:
merah (1), biru (2), hijau (3)
$(m = 3)$



Pewarnaan graf: pseudo-code backtracking (1)

```
Procedure mColouring(k:integer);
Var stop:Boolean;
Begin
    stop ← false
    while not stop do
       WarnaiSimpul(k)    {coba isi k dengan sebuah warna}
       if x[k] = 0 then   {semua warna telah dicoba}
          stop ← true
       else
          if k = n then   {semua simpul telah diberi warna}
             write(x[1],x[2],...,x[n])   {cetak solusi}
          else
             mColouring(k+1)   {tinjau simpul berikutnya}
          endif
       endif
    endwhile
End
```

Pewarnaan graf: pseudo-code backtracking (2)

```
Procedure WarnaiSimpul(k:integer);
Var stop, keluar:Boolean; j:integer
Begin
    stop ← false
    while not stop do
       x[k] ← (x[k]+1) mod (m+1)  {coba warnai simpul k}
       if x[k] = 0 then    {semua warna telah dicoba}
          stop ← true
       else
          for j ← 1 to n do   {cek simpul-simpul lain}
             if G[k,j] = n and (x[k]=x[j])then
                exit loop
             endif
          endfor
          if j = n+1 then   {semua simpul telah diperiksa}
             stop ← true
          endif
       endif
    endwhile
End
```

Kompleksitas backtracking untuk pewarnaan graf

- Procedure `WarnaiSimpul` dapat dipanggil sebanyak banyaknya simpul pada ruang solusi, yaitu
  - 1 simpul pada level 0
  - m simpul pada level 1
  - m^2 simpul pada level 2

    $\vdots$

  - m^n simpul pada level n

  yaitu

$$1 + m + m^2 + \ldots + m^n = \frac{m^{n+1} - 1}{m - 1} \text{simpul}$$

- Procedure `WarnaiSimpul` mempunyai kompleksitas $O(mn)$
- Jadi, kompleksitas algoritma backtracking untuk pewarnaan graf adalah $O(nm^n)$

Latihan (2)

Di suatu negara terdapat 7 buah stasiun televisi. Pemerintah menetapkan aturan bahwa dua stasiun yang berjarak tidak lebih dari 150 km tidak boleh beroperasi pada saluran frekuensi yang sama. Tabel di bawah ini memperlihatkan jarak (km) stasiun televisi satu sama lain. Berapa banyak minimum frekuensi yang berbeda yang diperlukan, yang menjamin tidak ada dua stasiun televisi berdekatan yang beroperasi pada frekuensi yang sama? Gunakan algoritma Backtracking untuk menyelesaikan masalah tersebut (bentuk pohon ruang statusnya!).

	1	2	3	4	5	6	7
1	-	85	145	200	50	100	230
2	85	-	125	175	100	160	175
3	145	125	-	100	200	250	160
4	200	175	100	-	210	220	180
5	50	100	200	210	-	100	235
6	100	160	250	220	100	-	120

Permasalahan jumlah nilai subhimpunan

Permasalahan sum of subsets

- Diberikan suatu himpunan yang berisi n bilangan bulat positif yang berbeda satu sama lain: $w_1, w_2, \ldots, w_n$, serta sebuah bilangan bulat positif m.
- Permasalahan: tentukan semua himpunan bagian dari himpunan tersebut yang total nilai anggota-anggotanya sama dengan m
- Contoh: diberikan $W = \{11, 13, 24, 7\}$ dan $m = 31$.
  Himpunan bagian yang memenuhi adalah $\{11, 13, 7\}$ dan $\{24, 7\}$.
- Catatan: permasalahan ini tidak selalu mempunyai solusi

State-space tree untuk jumlah nilai subhimpunan

- misalkan kita tinjau himpunan $W = \{w_1, w_2, \ldots, w_n\}$ dan bilangan bulat positif m
- Solusi permasalahan jumlah nilai subhimpunan dapat dinyatakan dengan tupel $\{x_1, x_2, \ldots, x_n\}$, dimana $x_i = 0$ jika w_i tidak masuk ke dalam himpunan bagian, dan $x_i = 1$ nika w_i masuk ke dalam himpunan bagian (yang menjadi solusi)
- Kita dapat mencari solusi dengan mengisi solusi per komponen untuk mendapatkan solusi parsial per tahapnya
- Artinya, pada setiap langkah ke-i, kita tinjau apakah w_i masuk ke dalam solusi atau tidak (i.e. apakah $x_i = 0$ atau $x_i = 1$)
- Sehingga, level pohon menyatakan urutan dari bilangan yang kita tinjau

Sum of subsets: contoh backtracking

Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:

start

$w_1 = 11$

$w_2 = 13$

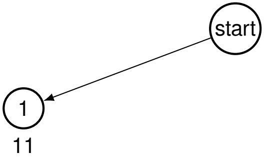
$w_3 = 24$

$w_4 = 7$

catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



$w_1 = 11$

$w_2 = 13$

$w_3 = 24$

$w_4 = 7$

catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

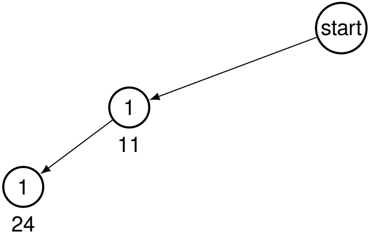
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:

$w_1 = 11$

$w_2 = 13$

$w_3 = 24$

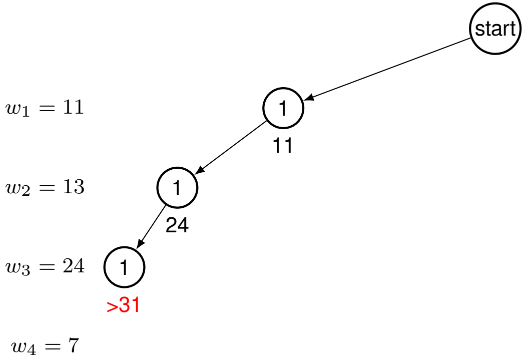
$w_4 = 7$



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

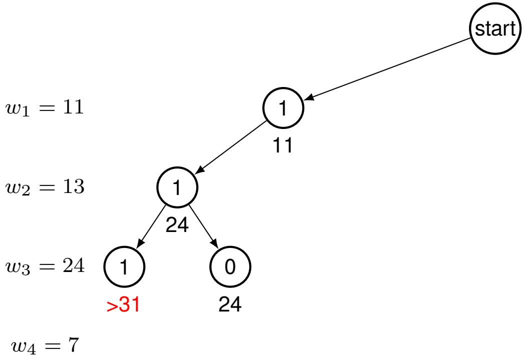
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

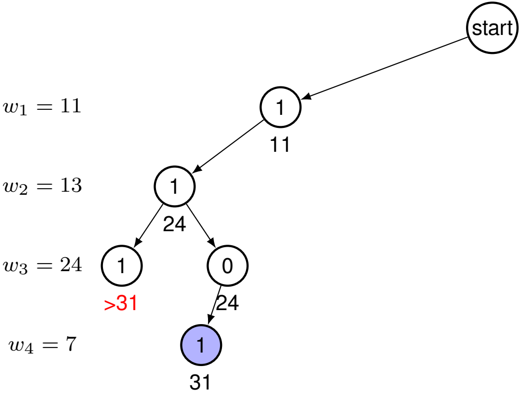
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

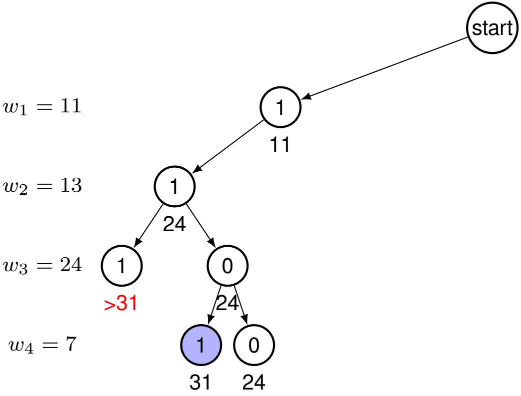
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

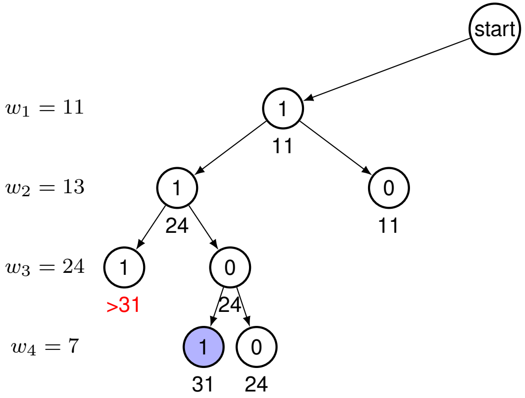
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

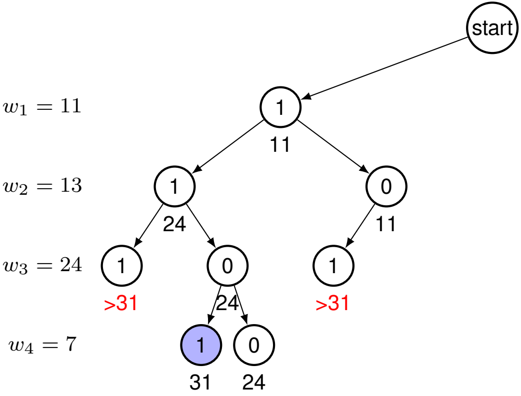
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

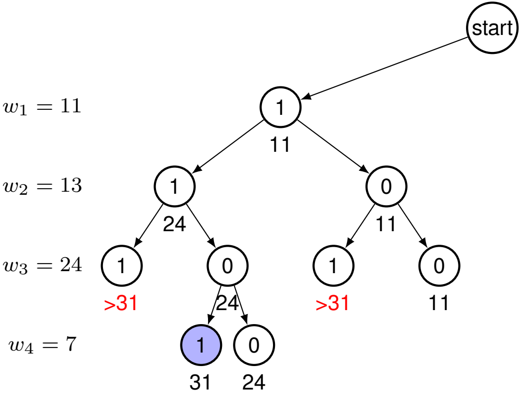
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

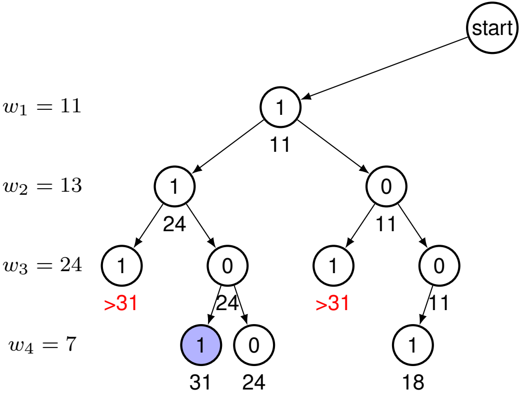
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

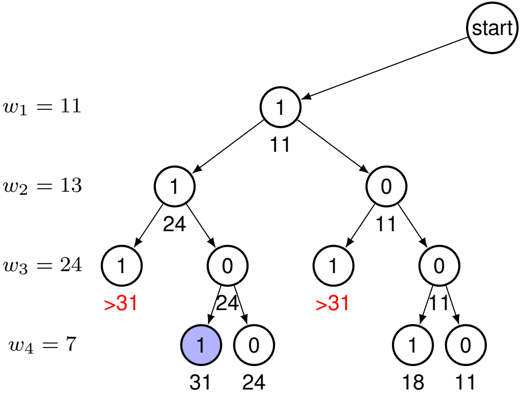
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

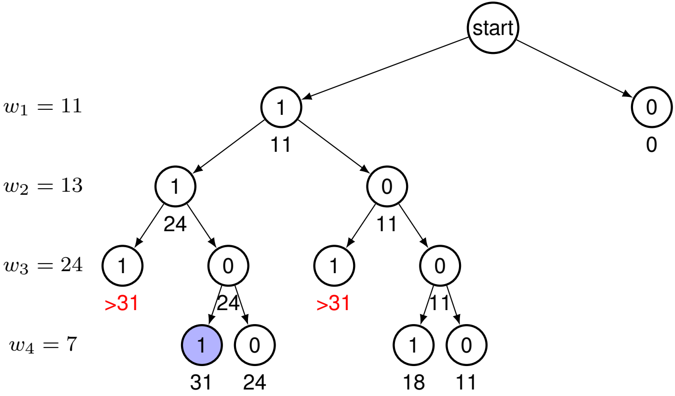
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

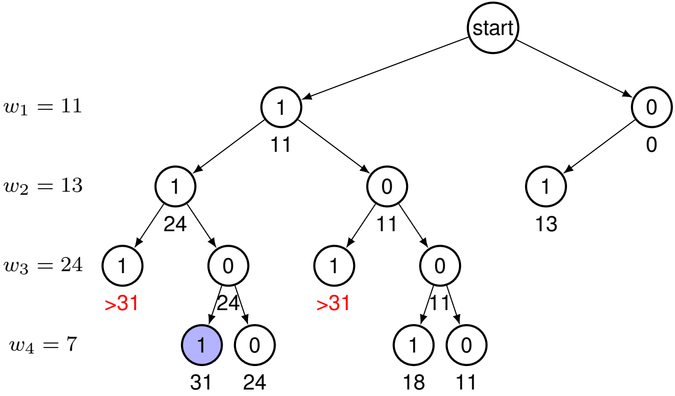
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

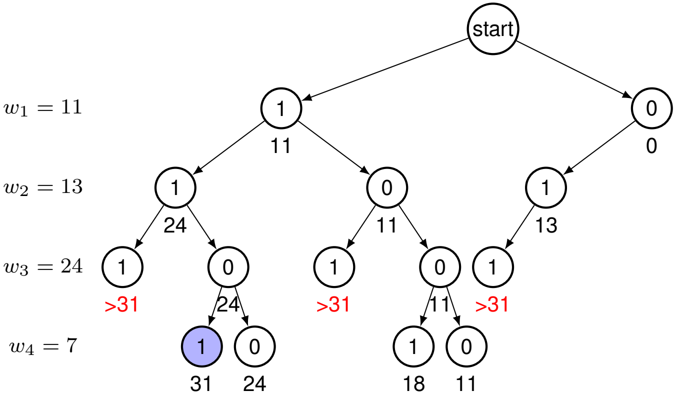
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

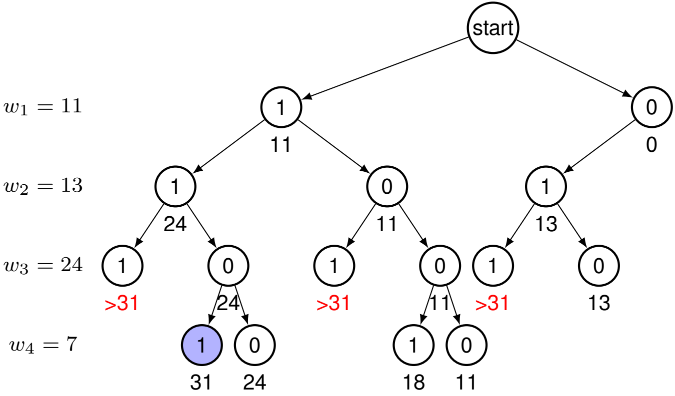
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

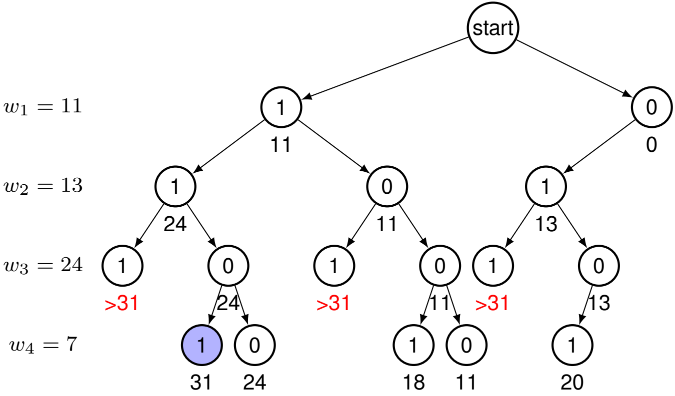
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

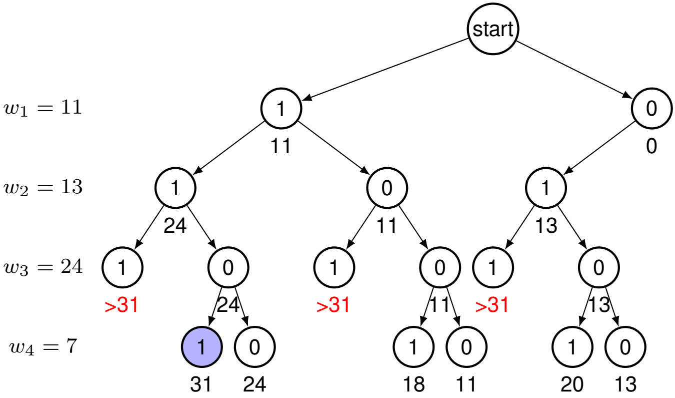
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

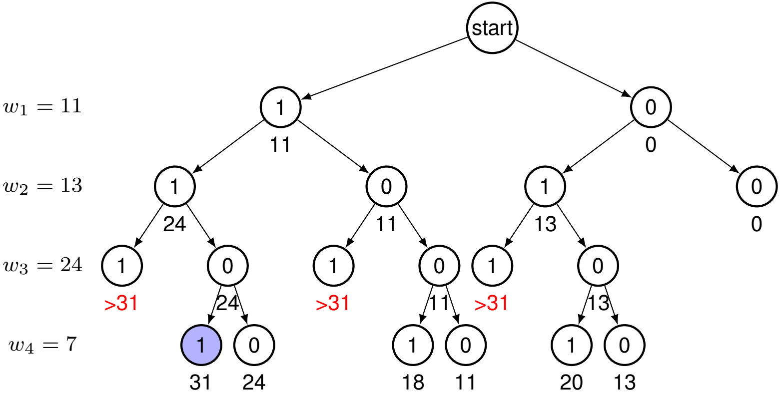
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

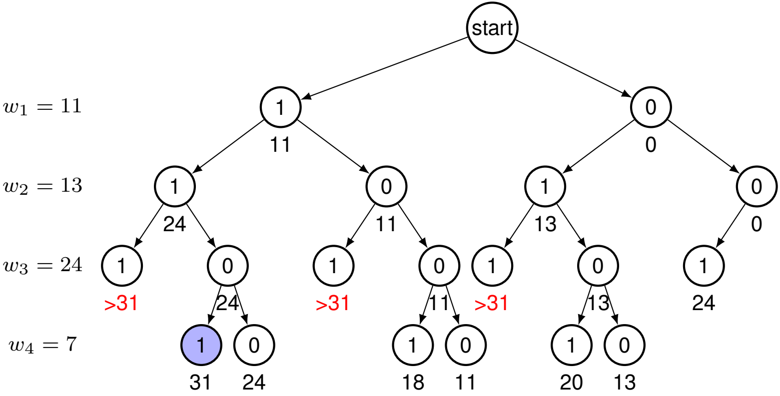
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

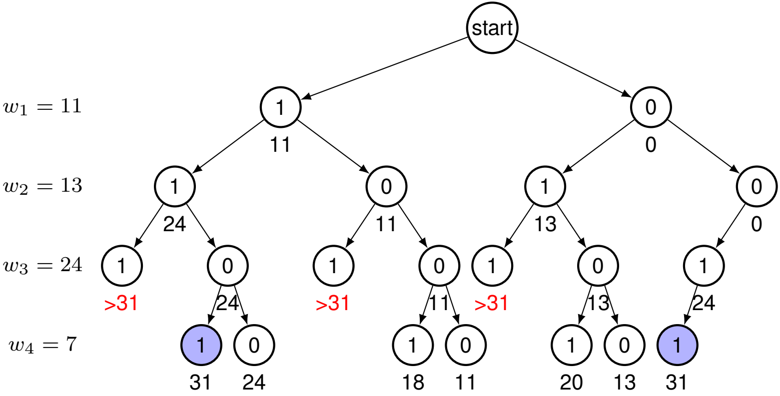
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

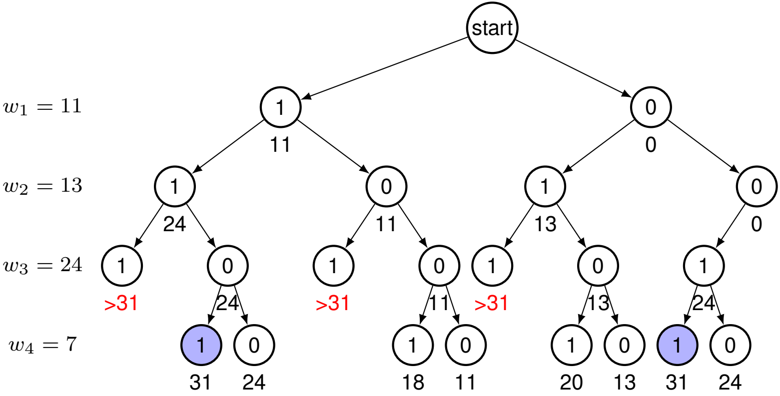
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

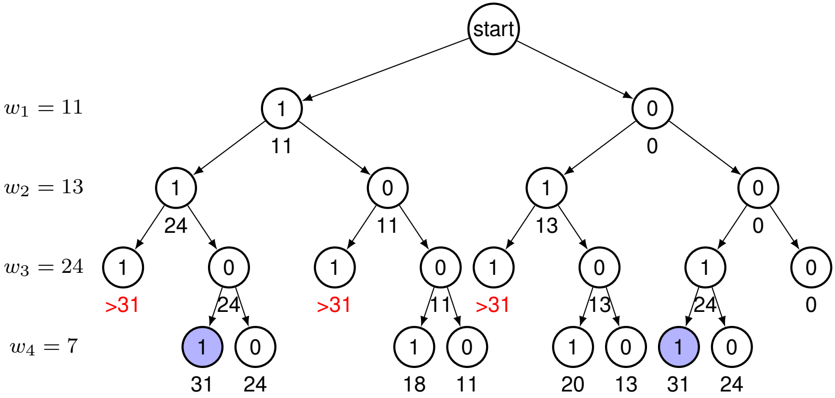
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

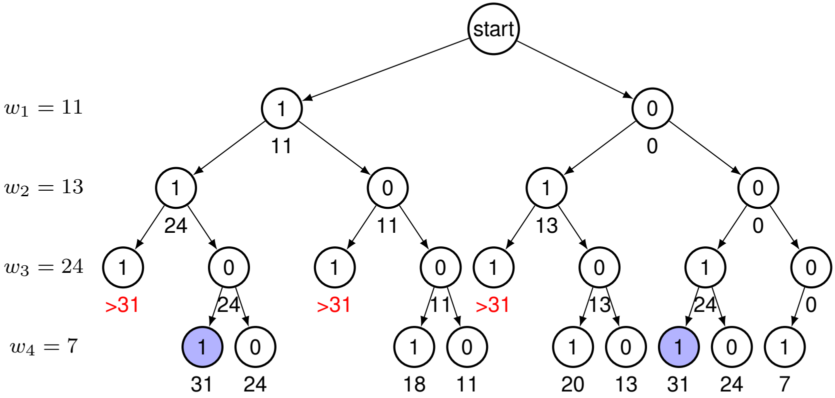
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

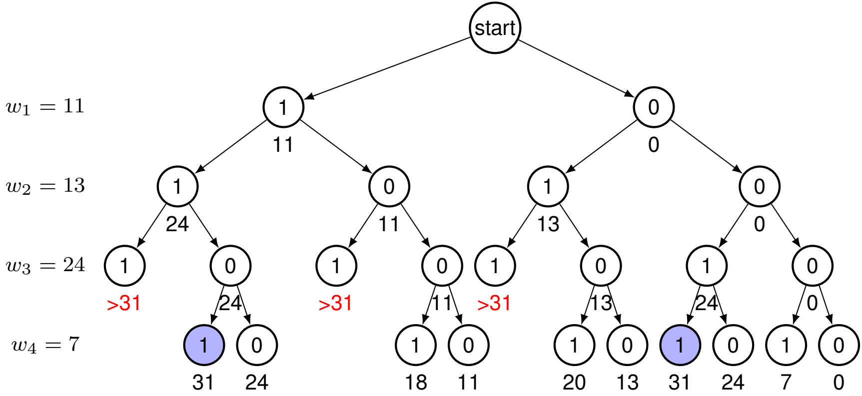
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Sum of subsets: contoh backtracking

Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: angka di bawah simpul menyatakan jumlah nilai bilangan yang telah terpilih (so far)

Perbaikan

- Kita dapat mengurutkan bilangan-bilangan dalam himpunan, secara menaik
- Misalkan *W* menyatakan jumlah nilai yang terpilih hingga tahap ke-i pada suatu solusi parsial yang terbentuk
- Solusi parsial tersebut tidak menjanjikan jika

  $$W \neq m \text{ dan } W + w_{i+1} > m$$

  karena dengan menambah bilangan lain, kita akan selalu mendapatkan bilangan yang lebih besar dari m
- Solusi parsial tersebut juga tidak menjanjikan jika

  $$W + sisa < m$$

  dimana sisa menyatakan total nilai bilangan yang belum ditinjau (karena walaupun kita pilih semuanya, kita tidak akan mencapai nilai m)

Sum of subsets: perbaikan backtracking

Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:

start

$w_1 = 7$

$w_2 = 11$

$w_3 = 13$

$w_4 = 24$

catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

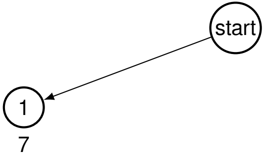
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:

$w_1 = 7$

$w_2 = 11$

$w_3 = 13$

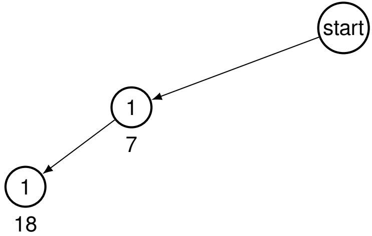
$w_4 = 24$



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



$w_1 = 7$

$w_2 = 11$

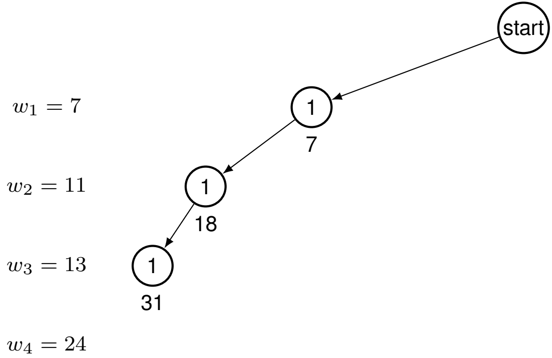
$w_3 = 13$

$w_4 = 24$

catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

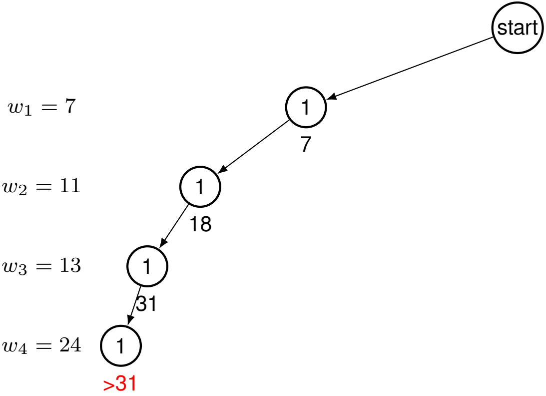
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

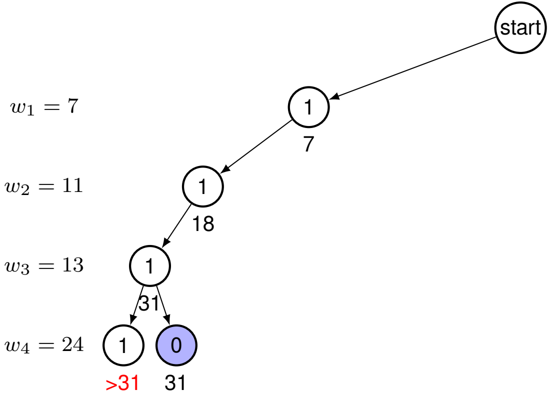
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

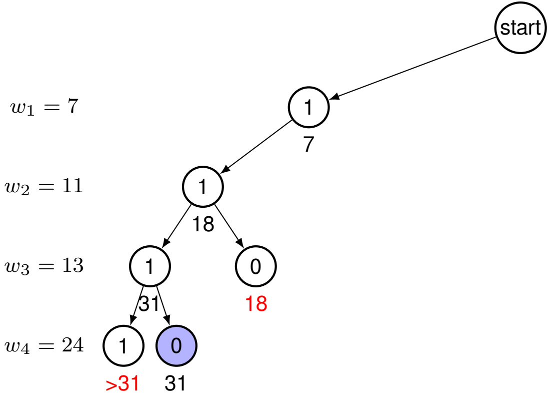
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

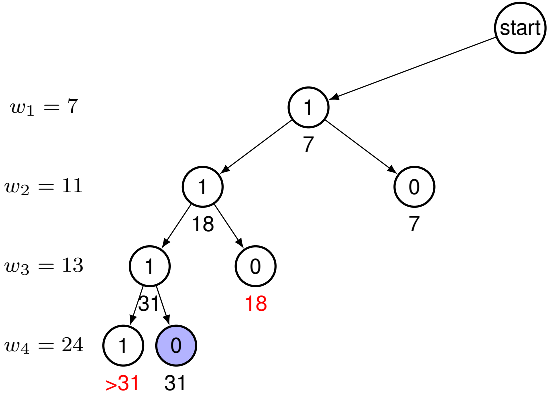
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

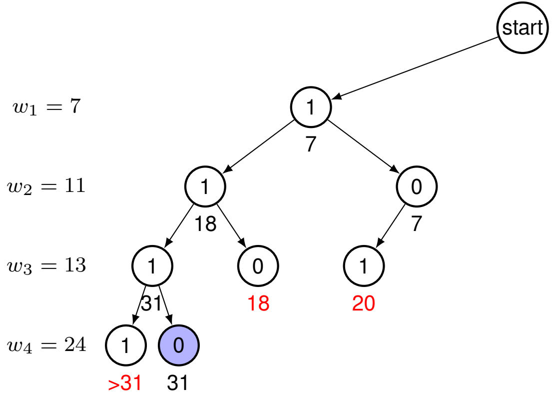
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

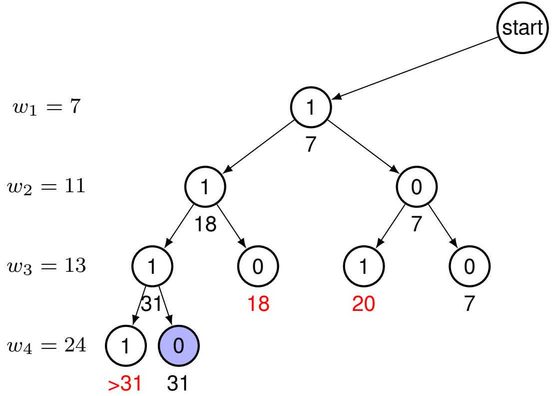
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

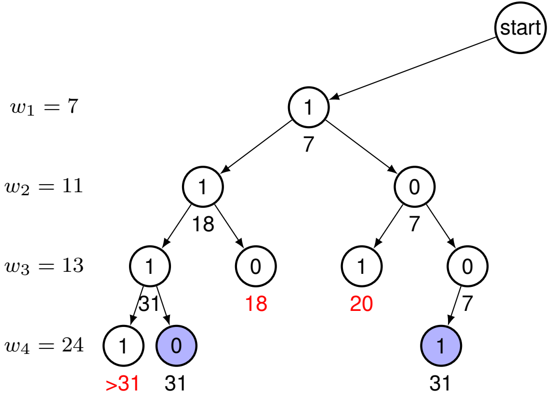
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

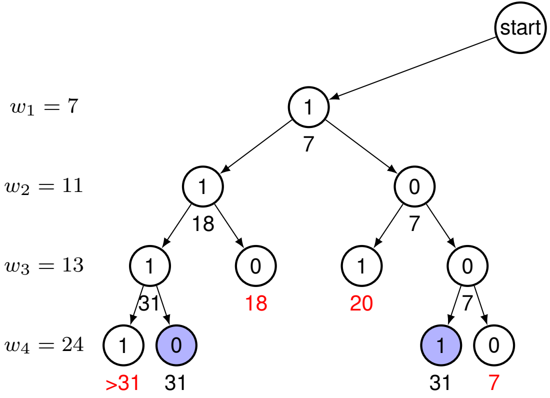
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

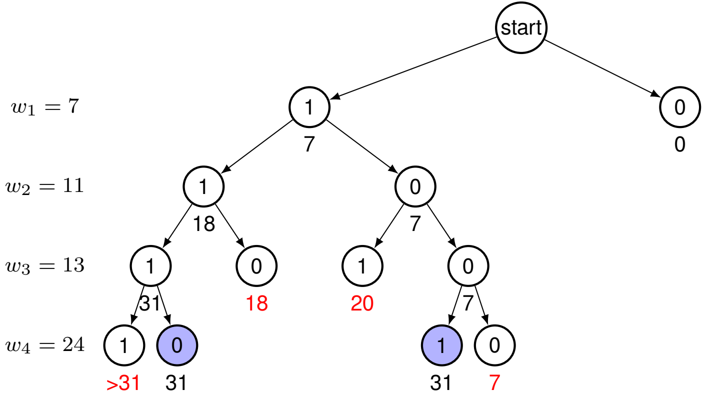
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

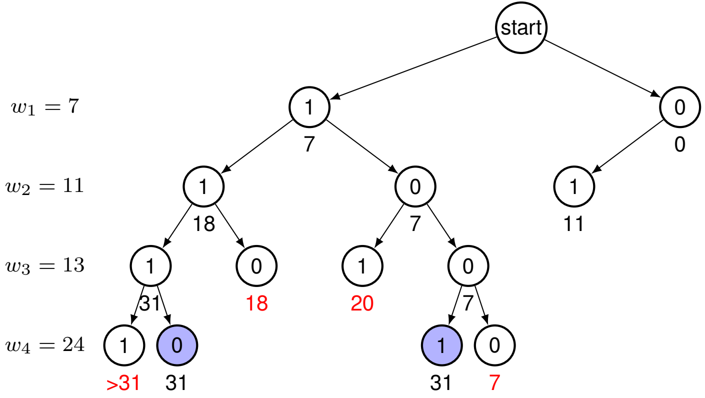
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

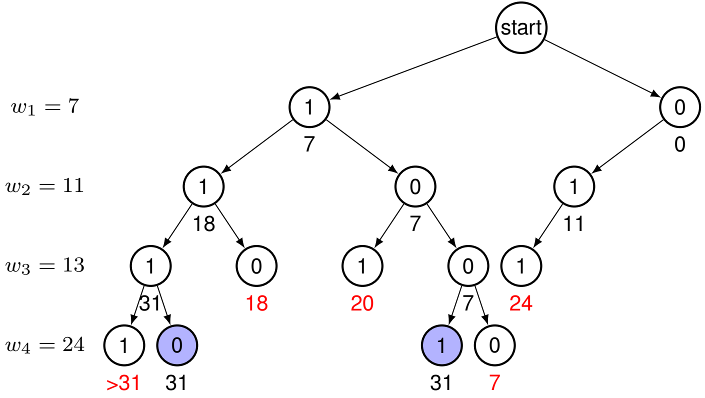
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

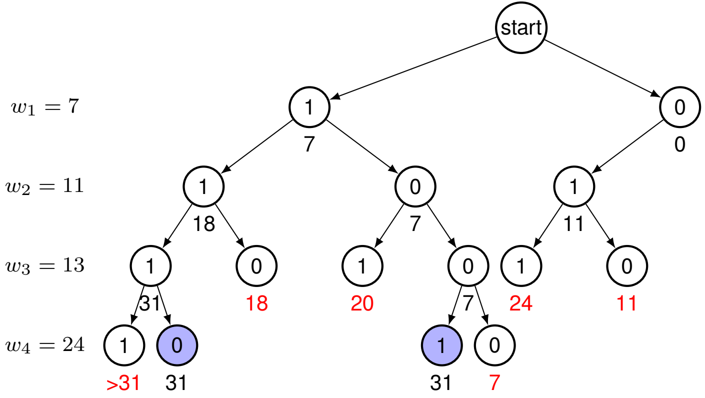
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

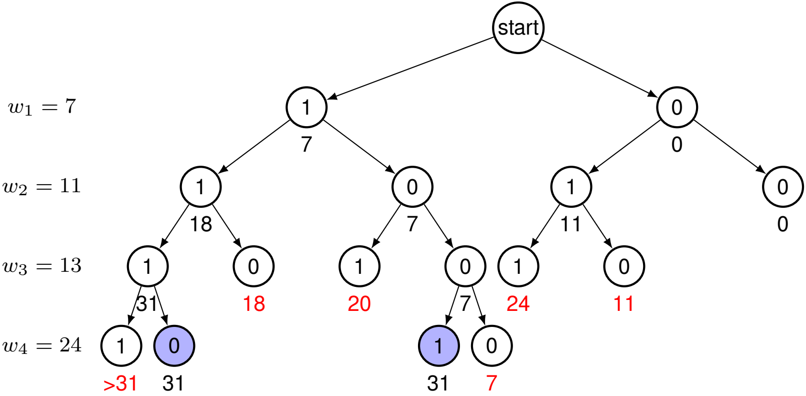
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

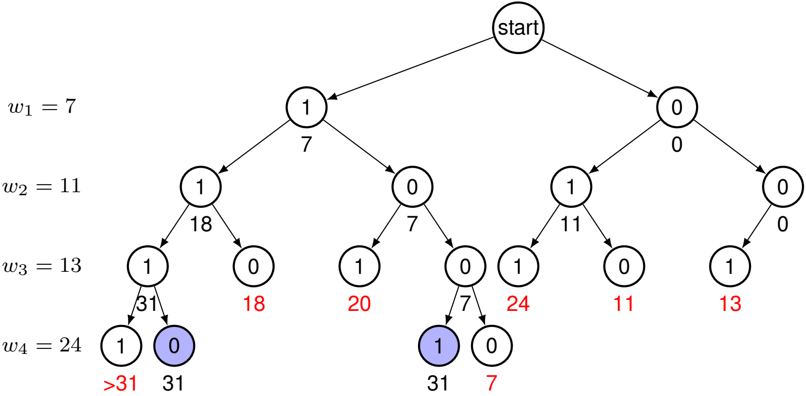
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

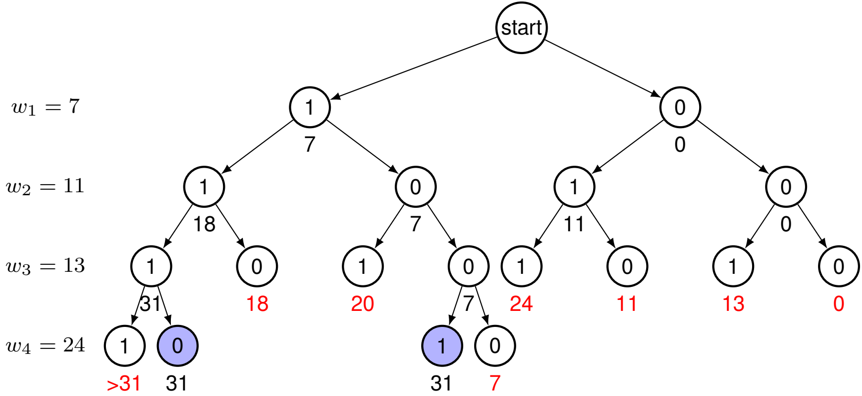
Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: perbaikan backtracking

Untuk $W = \{11, 13, 24, 7\}$ dan $m = 31$:



catatan: tidak menjanjikan jika $W \neq m$ dan $W + w_{i+1} > m$ ATAU $W + sisa < m$

Sum of subsets: pseudo-code backtracking

```
Procedure
SumOfSubsets(k:integer,W:integer,sisa:integer,m:integer);
 Begin
     if (W=m or W+w[k+1] ≤ m) and (W+sisa ≥ m) then
        if W=m then
           write(x[1],x[2],...,x[n])
        else
           x[k+1] ← 1
           SumOfSubsets(k+1,W+w[k+1],sisa-w[k+1],m)
           x[k+1] ← 0
           SumOfSubsets(k+1,W,sisa-w[k+1],m)
        endif
     endif
 End
```

Kompleksitas backtracking untuk sum of subsets

Procedure `SumOfSubsets()` dapat dipanggil sebanyak banyaknya simpul pada ruang solusi, yaitu:

- 1 simpul pada level 0
- 2 simpul pada level 1
- 2^2 simpul pada level 2

$$\vdots$$

- 2^n simpul pada level n

yaitu

$$1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1 \text{ simpul}$$

Latihan (3)

Diberikan $A = \{2, 10, 15, 27, 42\}$. Gunakan algoritma backtracking dan gambarkan pencarian ruang solusi (state space tree) untuk mencari salah satu himpunan bagian pertama dari A yang mempunyai jumlah 52!

Latihan (4)

Seorang petani yang harus menyeberang sungai dengan membawa serigala, domba, dan kubis. Perahu yang digunakan petani tersebut hanya bisa membawa dirinya dan salah satu dari barang bawaannya tersebut.

Jika ditinggal tanpa pengawasan petani, serigala akan memangsa domba, atau domba akan memakan kubis. Oleh karena itu, serigala dan domba, atau domba dan kubis, tidak dapat ditinggalkan bersama. Desainlah solusi dengan menerapkan strategi backtracking agar petani tersebut dapat menyeberangkan semuanya dengan selamat!

Referensi

1. T. H. Cormen, C. E. Leiserson, R. L. Riverst, C. Stein. Introduction to Algorithms –3rd Edition, MIT Press, 2009.
2. A. Levitin. Introduction to The Design and Analysis of Algorithms –3rd Edition, Pearson, 2011.
3. R. Neapolitan, K. Naimipour. Foundations of Algorithms –5th Edition, Jones and Bartlett Learning, 2014.
4. Ir. Rinaldi Munir, M.T. Diktat Strategi Algoritmik IF2251. Departemen Teknik Informatika, Institut Teknologi Bandung
5. Ian Parberry. Lecture notes on Algorithm Analysis. Department of Computer Sciences, University of North Texas, 2001

THANK YOU