Algoritma Runut-balik (*Backtracking*)

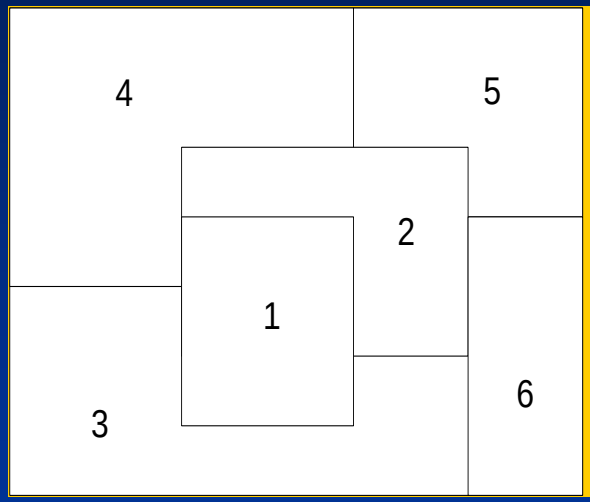
Bagian 2

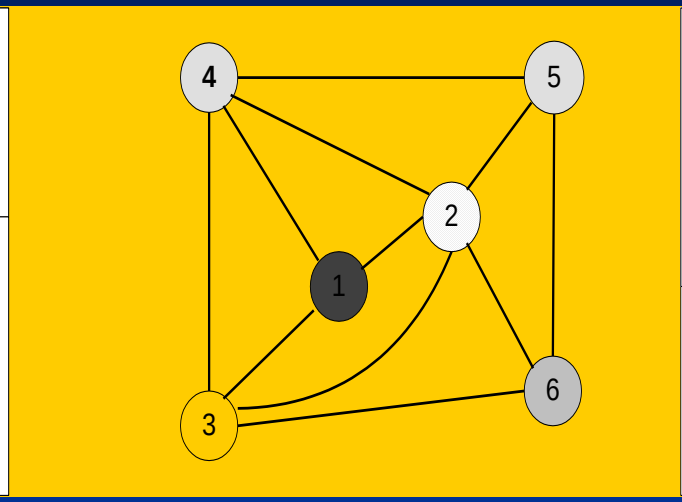
Pewarnaan Graf (*Graph Colouring*)

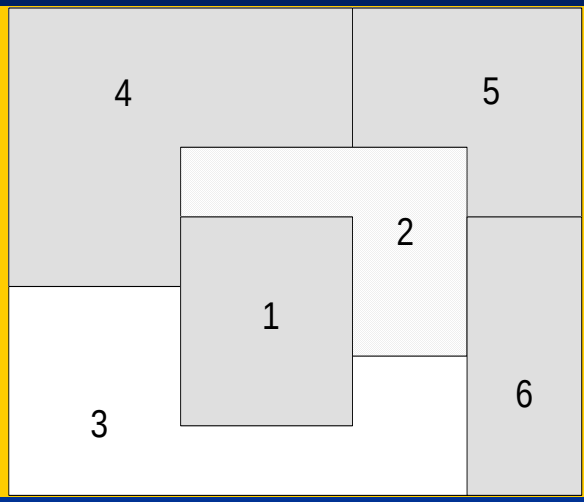
Persoalan:

- Diberikan sebuah graf *G* dengan *n* buah simpul dan disediakan *m* buah warna. Bagaimana mewarnai seluruh simpul graf *G* sedemikian sehingga tidak ada dua buah simpul bertetangga yang mempunyai warna sama (Perhatikan juga bahwa tidak seluruh warna harus dipakai)

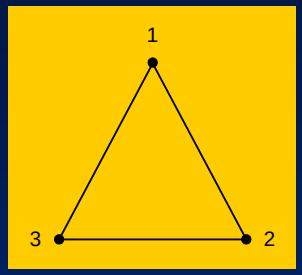
Contoh aplikasi: pewarnaan peta

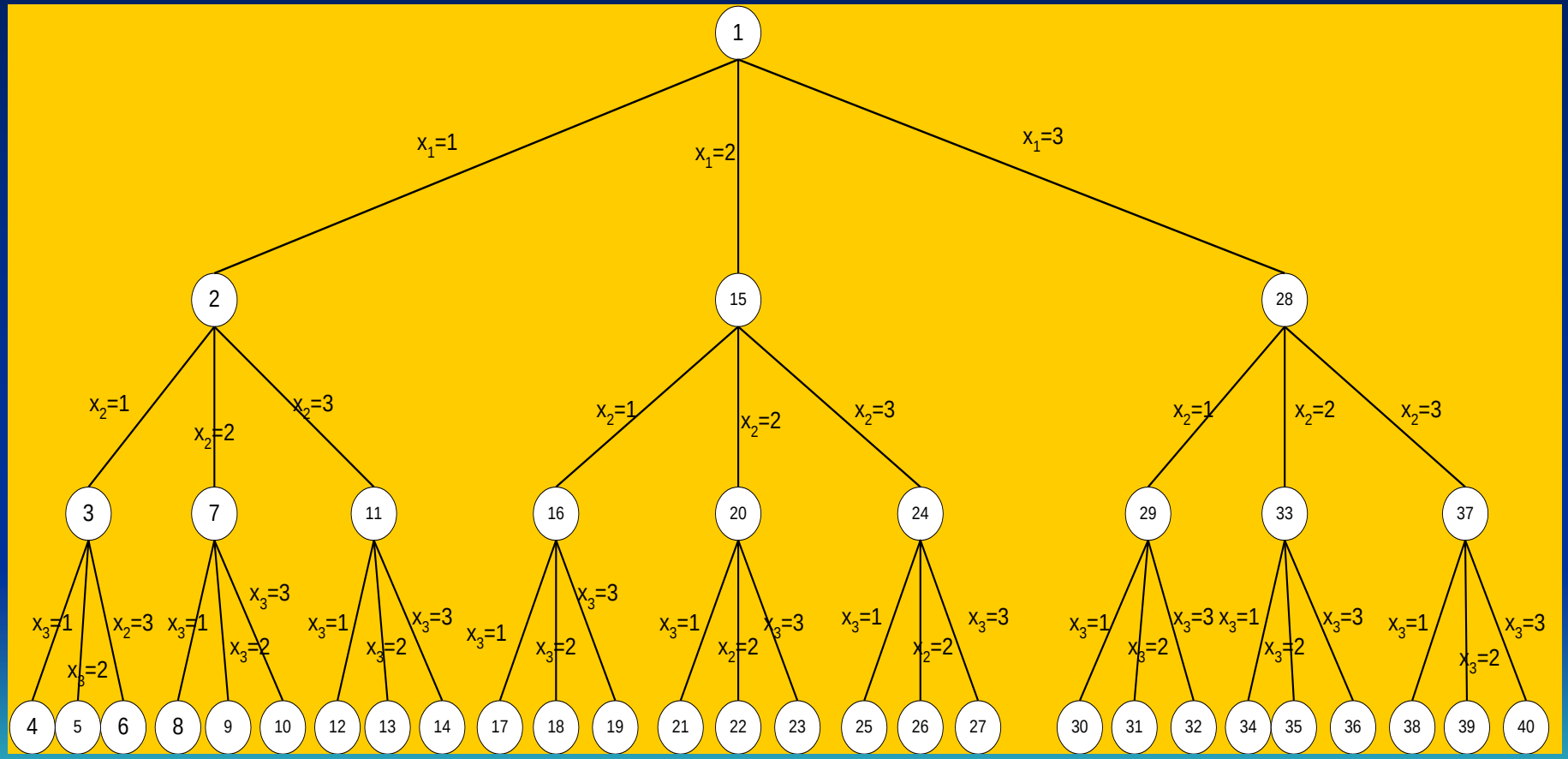






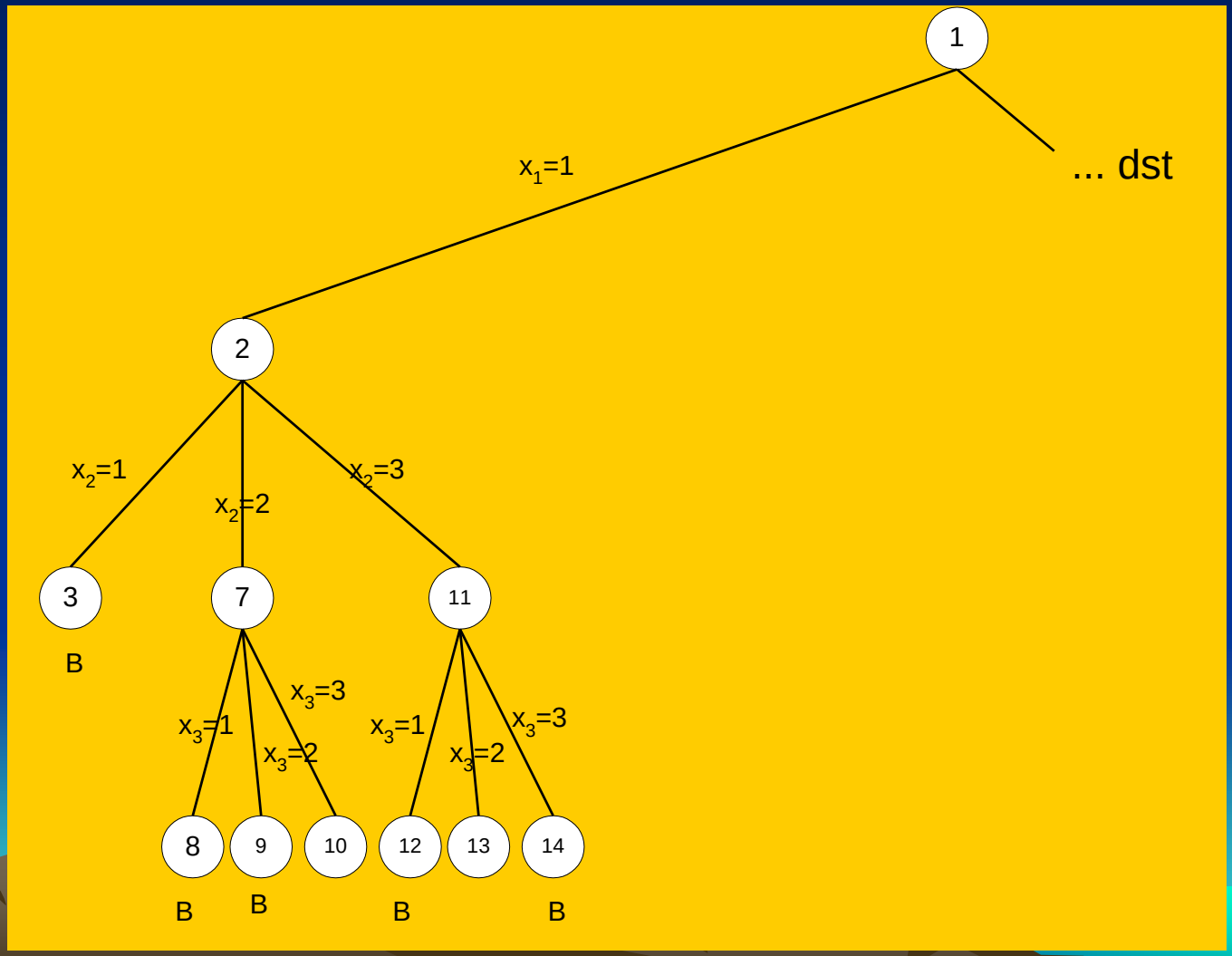
Tinjau untuk $n = 3$ dan $m = 3$.





Misalkan warna dinyatakan dengan angka 1, 2, …, *m* dan solusi dinyatakan sebagai vektor X dengan *n-tuple*:

$X = (x_1 , x_2 , ..., x_n) , \quad x_i \in \{ 1, 2, \ldots, m\}$



Algoritma Runut-balik Untuk Pewarnaan Graf

- Masukan:

  1. Matriks ketetanggan GRAF[1..n, 1..n]

     GRAF[i,j] = true jika ada sisi (i,j)

     GRAF[i,j] = false jika tidak ada sisi (i,j)

  2. Warna

     Dinyatakan dengan integer 1, 2, ...,m

- Keluaran:

  1. Tabel X[1..n], yang dalam hal ini, x[i] adalah warna untuk simpul i.

- Algoritma:

  1. Inisialisasi x[1..n] dengan 0

```
for i←1 to n do
  x[i]←0
endfor
```

2. Panggil prosedur PewarnaanGraf(1)

```
procedure PewarnaanGraf(input k : integer)
{ Mencari semua solusi solusi pewarnaan graf; rekursif
  Masukan: k adalah nomor simpul graf.
  Keluaran: jika solusi ditemukan, solusi dicetak ke piranti
keluaran
}
Deklarasi
  stop : boolean

Algoritma:
  stop←false
  while not stop do
     {tentukan semua nilai untuk x[k] }
     WarnaBerikutnya(k) {isi x[k] dengan sebuah warna}
     if x[k] = 0 then    {tidak ada warna lagi, habis}
      stop←true
     else
      if k=n then       {apakah seluruh simpul sudah diwarnai?}
         CetakSolusi(X,n)
      else
         PewarnaanGraf(k+1)   {warnai simpul berikutnya}
       endif
    endif
  endwhile
```

```
procedure WarnaBerikutnya(input k:integer)
{ Menentukan warna untuk simpul k

 Masukan: k
 Keluaran: nilai untuk x[k]

 K.Awal: x[1], x[2], ... , x[k-1] telah diisi dengan warna dalam
himpunan {1,2, …, m} sehingga setiap simpul bertetangga mempunyai
warna berbeda-beda.
 K.Akhir: x[k] berisi dengan warna berikutnya apabila berbeda dengan
warna simpul-simpul tetangganya. Jika tidak ada warna yang dapat
digunakan, x[k] diisi dengan nol
}
Deklarasi
  stop, keluar : boolean
  j : integer

Algoritma:
  stop←false
  while not stop do
     x[k]←(x[k]+1) mod (m+1)  {warna berikutnya}
     if x[k]=0  then          {semua warna telah terpakai}
      stop←true
     else
        {periksa warna simpul-simpul tetangganya}
       j←1
       keluar←false
        while (j≤n) and (not keluar) do

           if (GRAF[k,j]) {jika ada sisi dari simpul k ke simpul j}
               and              {dan}
            (x[k] = x[j])      {warna simpul k = warna simpul j }
           then
            keluar←true {keluar dari kalang}
           else
            j←j+1       {periksa simpul berikutnya}
          endif
        endwhile
        { j > n or keluar}

        if j=n+1 {seluruh simpul tetangga telah diperiksa dan
                  ternyata warnanya berbeda dengan x[k] }
        then
           stop←true    {x[k] sudah benar, keluar dari kalang}
        endif

     endif
  endwhile
```

Kompleksitas Waktu algoritma PewarnaanGraf

- Pohon ruang status yang untuk persoalan pewarnaan graf dengan *n* simpul dan *m* warna adalah pohon *m-ary* dengan tinggi *n* + 1.

- Tiap simpul pada aras *i* mempunyai *m* anak, yang bersesuaian dengan *m* kemungkinan pengisian $x[i]$, $1 \leq i \leq n$.

- Simpul pada aras $n+1$ adalah simpul daun. Jumlah simpul internal (simpul bukan daun) ialah $\sum_{i=0}^{n-1} m^i$.

- Tiap simpul internal menyatakan pemanggilan prosedur `WarnaBerikutnya` yang membutuhkan waktu dalam $O(mn)$. Total kebutuhan waktu algoritma `PewarnaanGraf` adalah

$$\sum_{i=1}^{n} m^i n = \frac{n(m^{n+1}-1)}{(m-1)} = O(nm^n)$$