

# P VS NP PROBLEMS

# Sebuah persoalan dikatakan...

- Solvable, jika terdapat mesin Turing yang dapat menyelesaikannya.  
(atau *decidable* pada jenis persoalan keputusan)
- *Unsolvable*, jika tidak dapat dibuat mesin Turing untuk menyelesaikannya.  
(atau *undecidable* pada jenis persoalan keputusan)
- *Solvable problem* dapat dibagi menjadi dua kategori:
  1. *Tractable*
  2. *Intractable*

- Sebuah persoalan dikatakan *tractable* jika ia dapat diselesaikan dalam waktu yang wajar (*reasonable*).
- Sebuah persoalan dikatakan *intractable* jika ia tidak dapat diselesaikan dalam waktu yang wajar dengan bertambahkannya ukuran persoalan.
- Apa yang dimaksud dengan waktu yang wajar? Standar waktunya adalah *polynomial time*.
  - *Polynomial time*:  $O(n^2)$ ,  $O(n^3)$ ,  $O(1)$ ,  $O(n \lg n)$
  - *Not in polynomial time*:  $O(2^n)$ ,  $O(n^n)$ ,  $O(n!)$  untuk  $n$  yang kecil

# Kompleksitas Waktu

- Mesin Turing adalah model matematis sebuah komputer.
- Persoalan dipecahkan dengan menggunakan mesin Turing.
- Mesin Turing menerima input sepanjang  $n$ , lalu, dimulai dari status awal, melakukan gerakan (transisi) dari satu status ke status lainnya.
- Kebutuhan waktu (*running time*) yang diperlukan oleh Mesin Turing untuk menyelesaikan persoalan dengan input dengan panjang  $n$ , dinyatakan dengan  $T(n)$ , disebut kompleksitas waktu.

- Sebuah mesin Turing  $M$  dikatakan kompleksitas waktunya  $T(n)$  bilamana jika  $M$  diberi input  $w$  yang panjangnya  $n$  maka  $M$  berhenti setelah melakukan gerakan sebanyak  $T(n)$ , tanpa memperhatikan apakah  $M$  berhenti pada status *accept* atau *reject*.

Contoh:  $T(n) = 5n + 2$   
 $T(n) = 2n^2 + n + 6$   
 $T(n) = 3^n + 10n^3$

# Kelas P

- Sebuah persoalan keputusan dikatakan di dalam kelas P jika ia dapat diselesaikan dalam waktu polinomial oleh mesin Turing deterministik.
- Dengan kata lain, terdapat mesin Turing deterministik  $M$  yang memecahkan persoalan dan berhenti pada status akhir dengan jumlah gerakan tidak lebih dari  $T(n)$  bilamana mesin Turing  $M$  diberikan input sepanjang  $n$ .
- Karena persoalan berkoresponden dengan bahasa, maka kita katakan bahasa  $L$  termasuk dalam kelas  $P$  jika terdapat polinomial  $T(n)$  sedemikian sehingga  $L = L(M)$  untuk mesin Turing deterministik  $M$  yang kompleksitas waktunya  $T(n)$

- Contoh persoalan dalam kelas P:
  1. Menentukan apakah sebuah *integer*  $n$  bilangan prima.
  2. Menentukan apakah sebuah *integer*  $x$  terdapat di dalam sebuah senarai (*list*).
  3. Menentukan apakah sebuah graf berbobot mengandung *minimum spanning tree* dengan bobot  $\leq W$ ?
  4. Menentukan apakah sebuah *integer*  $n$  merupakan elemen terbesar di dalam sebuah senarai?
- Persoalan di dalam kelas P disebut ***tractable*** sedangkan persoalan yang bukan di dalam P disebut ***intractable***.

# Mesin Turing Deterministik vs. Mesin Turing non Deterministik

- Pada mesin Turing deterministik, untuk satu status ( $p$ ) dan satu simbol ( $X_i$ ) di pita, hanya ada satu transisi ke status berikutnya ( $q$ ).

$$\delta(p, X_i) = (q, Y, L)$$

- Sebaliknya pada mesin Turing non-deterministik, untuk satu status ( $p$ ) dan satu simbol ( $X_i$ ) di pita, terdapat lebih dari satu transisi ke status berikutnya ( $q$ ). Mesin Turing non-deterministik memiliki kemampuan untuk memilih suatu transisi

$$\delta(p, X_i) = (q, W, L), \delta(p, X_i) = (r, Y, L), \delta(p, X_i) = (s, Z, R),$$



# Kelas NP

- Sebuah persoalan keputusan dikatakan di dalam kelas NP jika ia dapat diselesaikan dalam waktu polinomial oleh mesin Turing non deterministik.
- Karena persoalan berkoresponden dengan bahasa, maka kita katakan bahasa  $L$  termasuk dalam kelas  $NP$  jika terdapat mesin Turing non deterministik dan kompleksitas waktu polinomial  $T(n)$  sedemikian sehingga  $L = L(M)$ , dan bilamana  $M$  diberikan input sepanjang  $n$ , maka jumlah gerakannya tidak lebih dari  $T(n)$ .

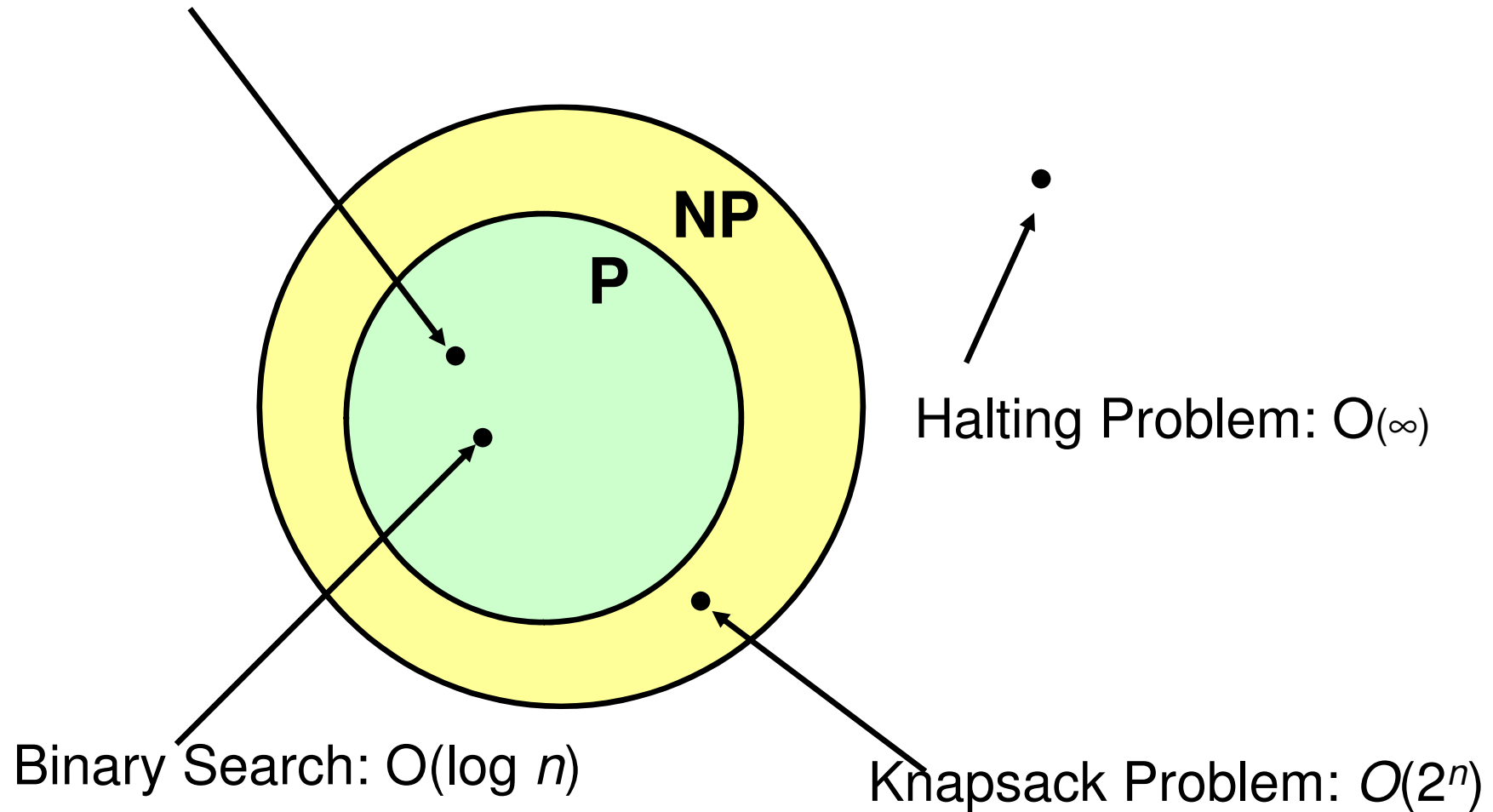
- Mesin Turing non deterministik dikatakan memiliki kebutuhan waktu polinomial karena ia memiliki kemampuan untuk menerka kemungkinan solusi yang jumlahnya eksponensial dan memverifikasi setiap kemungkinan solusi tersebut dalam waktu polinomial.
- Setiap mesin Turing deterministik adalah mesin Turing non deterministik, tetapi mesin Turing deterministik tidak memiliki kemampuan memilih transisi.
- Oleh karena itu,  $P \subseteq NP$ .
- Meskipun demikian, NP mengandung beberapa persoalan yang tidak termasuk ke dalam P.

Contoh persoalan dalam kelas NP:

1. *Travelling Salesperson Decision Problem (TSDP)*:  
Menentukan apakah sebuah graf berbobot memiliki tur terpendek yang melalui setiap simpul tepat sekali dan kembali ke simpul awal dengan bobot  $\leq M$ ?
2. *Integer Knapsack Decision Problem*: apakah dapat memasukkan objek-objek ke dalam *knapsack* namun tidak melebihi  $W$  tetapi total profitnya paling sedikit sebesar  $P$ .
3. *Graph-Colouring Decision Problem*: apakah terdapat pewarnaan graf yang menggunakan paling banyak  $m$  warna sedemikian sehingga dua simpul bertetangga memiliki warna berbeda?

# Kelas Persoalan

Algoritma Prim:  $O(n^2)$



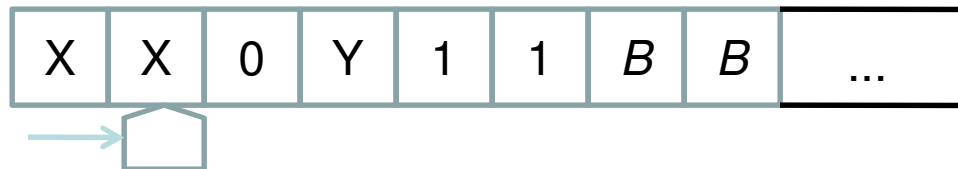
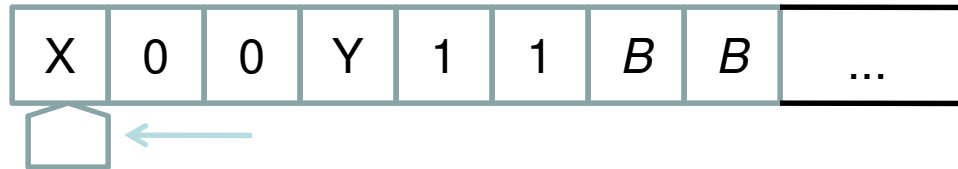
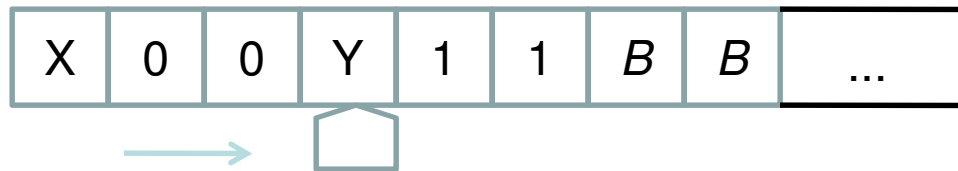
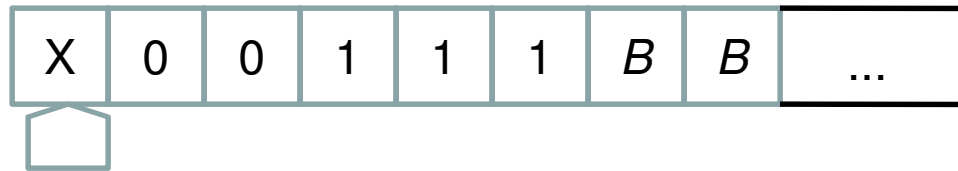
# Contoh Persoalan P

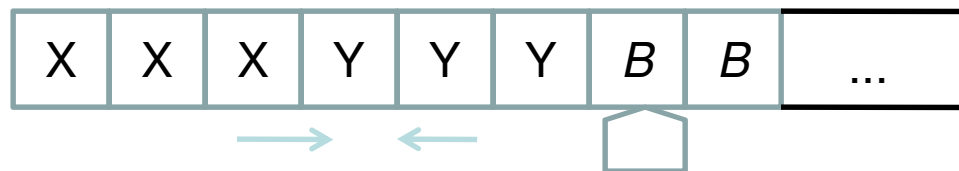
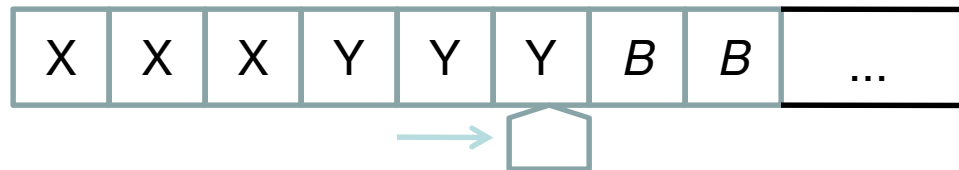
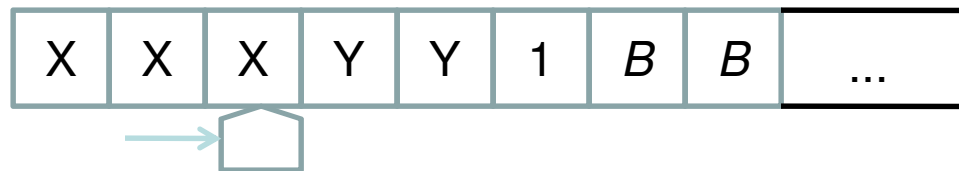
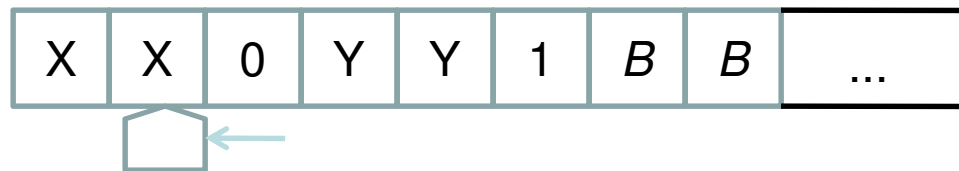
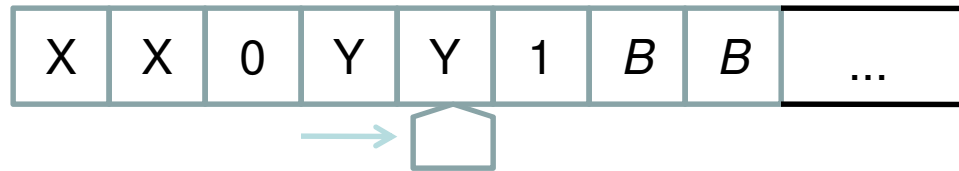
Mesin Turing  $M$  akan digunakan untuk mengenali bahasa  $L = \{0^n 1^n \mid n \geq 1\}$ . Diberikan string  $w$ , apakah  $w \in L$ ?

Algoritma:

1. Ganti simbol '0' paling kiri dengan simbol 'X'.
2. Gerakkan *head* ke kanan hingga dijumpai simbol '1'.
3. Ganti simbol '1' paling kiri dengan simbol 'Y'
4. Gerakkan *head* ke kiri hingga dijumpai simbol 'X'
5. Geser *head* ke kanan (akan diperoleh '0' paling kiri).
6. Kembali ke langkah 1.

Contoh:  $w = 000111$





Kesimpulan: string '000111' dikenali oleh *M*.

## Analisis kompleksitas:

1. Ganti simbol '0' paling kiri dengan simbol 'X'.  $\rightarrow O(1)$
2. Gerakkan *head* ke kanan hingga dijumpai simbol '1'  $\rightarrow O(n)$
3. Ganti simbol '1' paling kiri dengan simbol 'Y'  $\rightarrow O(1)$
4. Gerakkan *head* ke kiri hingga dijumpai simbol 'X'  $\rightarrow O(n)$
5. Geser *head* ke kanan (akan diperoleh '0' paling kiri).  $\rightarrow O(1)$
6. Kembali ke langkah 1.  $\rightarrow$  paling banyak  $n/2$  kali

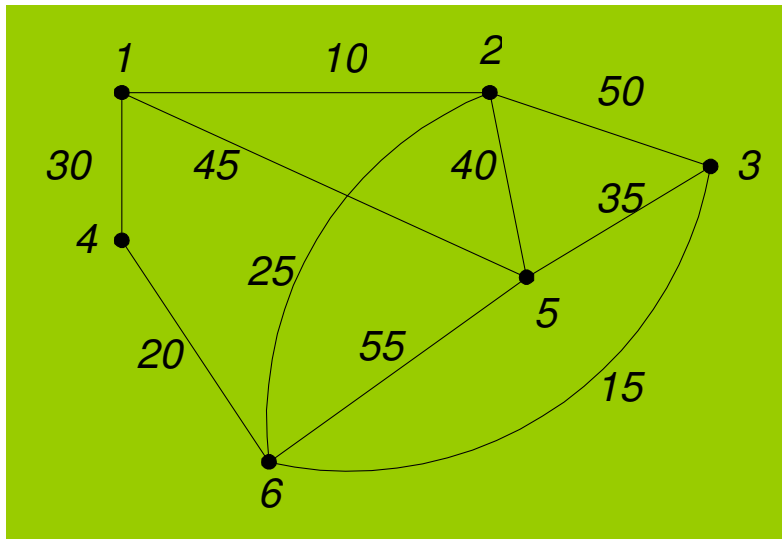
## Total kebutuhan waktu:

$$n/2 \{ O(1) + O(n) + O(1) + O(n) + O(1) \} = n/2 \{ O(n) \} = O(n^2)$$

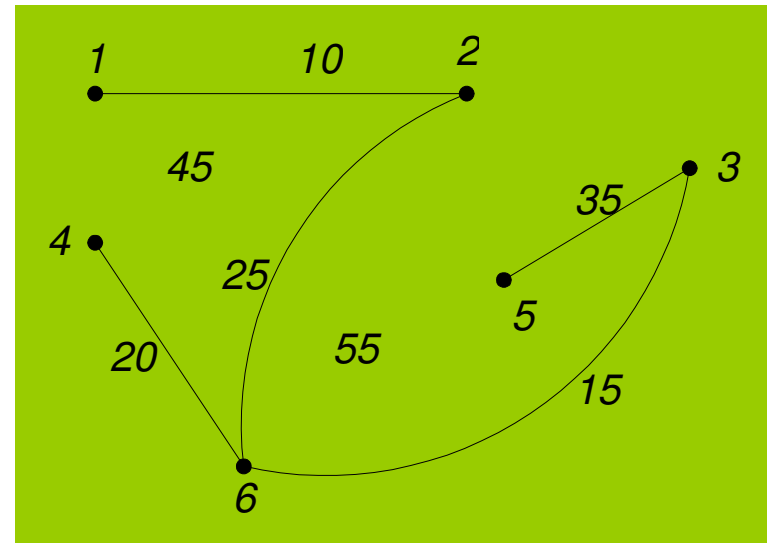


# Algoritma Prim

- Algoritma untuk menentukan pohon merentang minimum (*minimum spanning tree*).
- Algoritma polinomial, karena kebutuhan waktunya  $O(n^2)$ .



(a) Graf  $G = (V, E)$



(b) Pohon merentang minimum

## Algoritma Prim

Langkah 1: ambil sisi dari graf  $G$  yang berbobot minimum, masukkan ke dalam  $T$ .

Langkah 2: pilih sisi  $(u, v)$  yang mempunyai bobot minimum dan bersisian dengan simpul di  $T$ , tetapi  $(u, v)$  tidak membentuk sirkuit di  $T$ . Masukkan  $(u, v)$  ke dalam  $T$ .

Langkah 3: ulangi langkah 2 sebanyak  $n - 2$  kali.

```

procedure Prim(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung-
berbobot G.
Masukan: graf-berbobot terhubung  $G = (V, E)$ , dengan  $|V| = n$ 
Keluaran: pohon rentang minimum  $T = (V, E')$ 
}

```

#### Deklarasi

$i, p, q, u, v$  : integer

#### Algoritma

Cari sisi  $(p, q)$  dari  $E$  yang berbobot terkecil

$T \leftarrow \{(p, q)\}$

for  $i \leftarrow 1$  to  $n-2$  do

    Pilih sisi  $(u, v)$  dari  $E$  yang bobotnya terkecil namun  
    bersisian dengan simpul di  $T$

$T \leftarrow T \cup \{(u, v)\}$

endfor

Mencari sisi  $(p, q)$  dari  $E$  berbobot terkecil  $\rightarrow O(n)$

$T \leftarrow \{(p, q)\} \rightarrow O(1)$

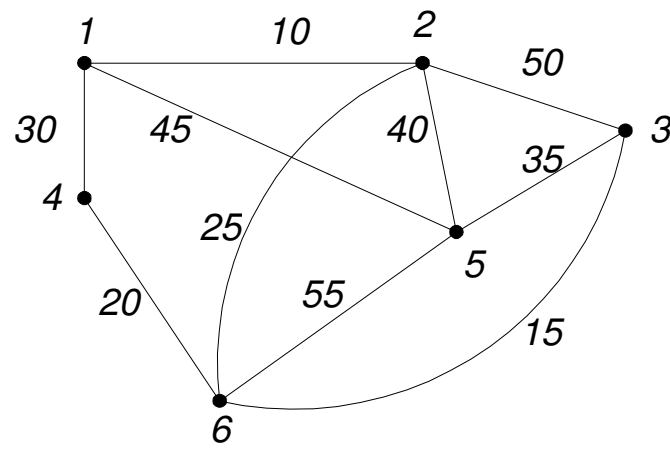
Pilih sisi  $(u, v)$  dari  $E$  berbobot terkecil namun bersisian dengan  $T \rightarrow O(n)$

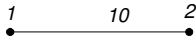
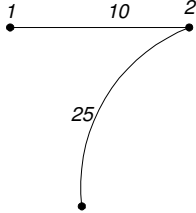
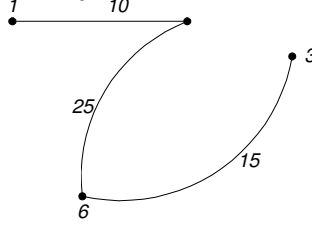
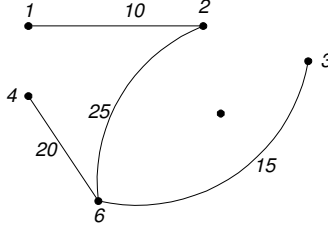
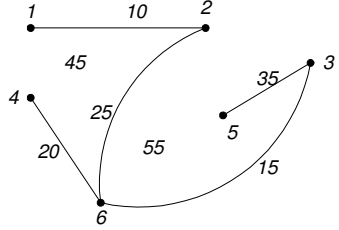
$T \leftarrow T \cup \{(u, v)\} \rightarrow O(1)$

Jumlah pengulangan loop  $\rightarrow n - 2$  kali

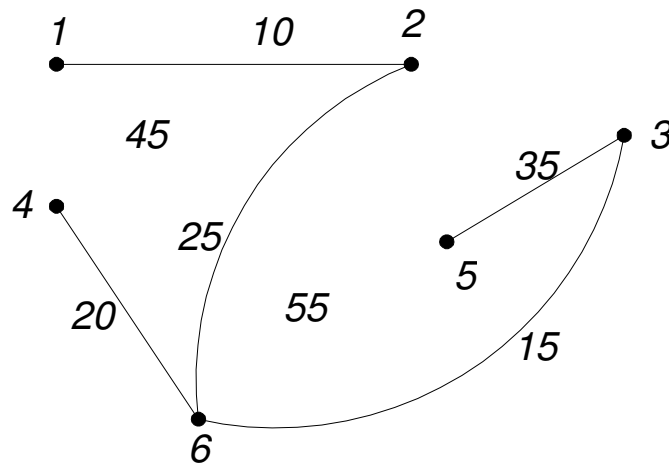
Total kebutuhan waktu  $\rightarrow O(n) + O(1) + (n - 2) \{ O(n) + O(1) \} = O(n) + O(n^2) = O(n^2)$

Contoh:



Langkah	Sisi	Bobot	Pohon rentang
1	(1, 2)	10	
2	(2, 6)	25	
3	(3, 6)	15	
4	(4, 6)	20	
5	(3, 5)	35	

Pohon merentang minimum yang dihasilkan:



$$\text{Bobot} = 10 + 25 + 15 + 20 + 35 = 105$$

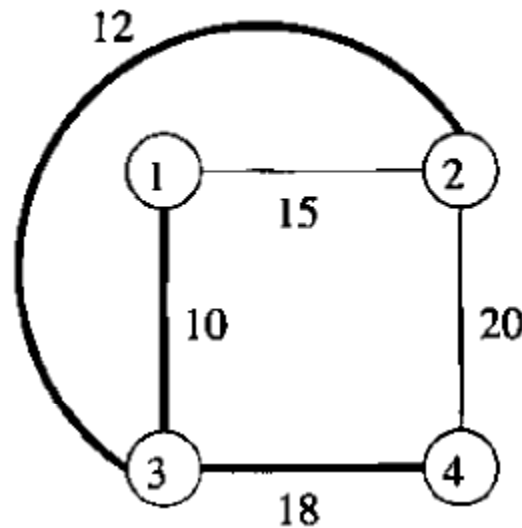
- Algoritma MST lainnya: Algoritma Kruskal dengan kompleksitas  $O(m \log m)$ ,  $m$  adalah jumlah sisi di dalam graf.
- Karena kita berhubungan dengan mesin Turing, maka kita memikirkan persoalan MST sebagai sebuah bahasa.
- Persolan keputusan  $\rightarrow$  Bahasa
- Jika MST dijadikan persoalan keputusan, maka deskripsinya adalah sbb: Diberikan sebuah graf berbobot  $G = (V, E)$  dan nilai  $W$ . Apakah  $G$  memiliki *spanning tree* dengan bobot  $\leq W$ ?

Pengkodean *MST-decision problem* pada mesin Turing:

1. Simpul-simpul diberi nomor 1 sampai  $m$ .
2. Setiap integer dikodekan dalam biner.
3. Kode dimulai dengan  $m$  dalam biner dan bobot  $W$  dalam biner, dipisahkan dengan koma.
4. Jika terdapat sisi dari simpul  $i$  dan  $j$  dengan bobot  $w$ , letakkan  $(i, j, w)$  di dalam kode. Integer dikodekan dalam biner.
5. Kode yang dihasilkan merupakan input di pita untuk mesin Turing deterministik  $M$ .



- Contoh: Diberikan graf di bawah ini dan  $W = 40$ .



- Kode untuk *MST-decision problem* di atas adalah:  
 100, 101000(1, 10, 1111)(1, 11, 1010)(10, 11, 1100)(10,  
 100, 10100)(11, 100, 10010)