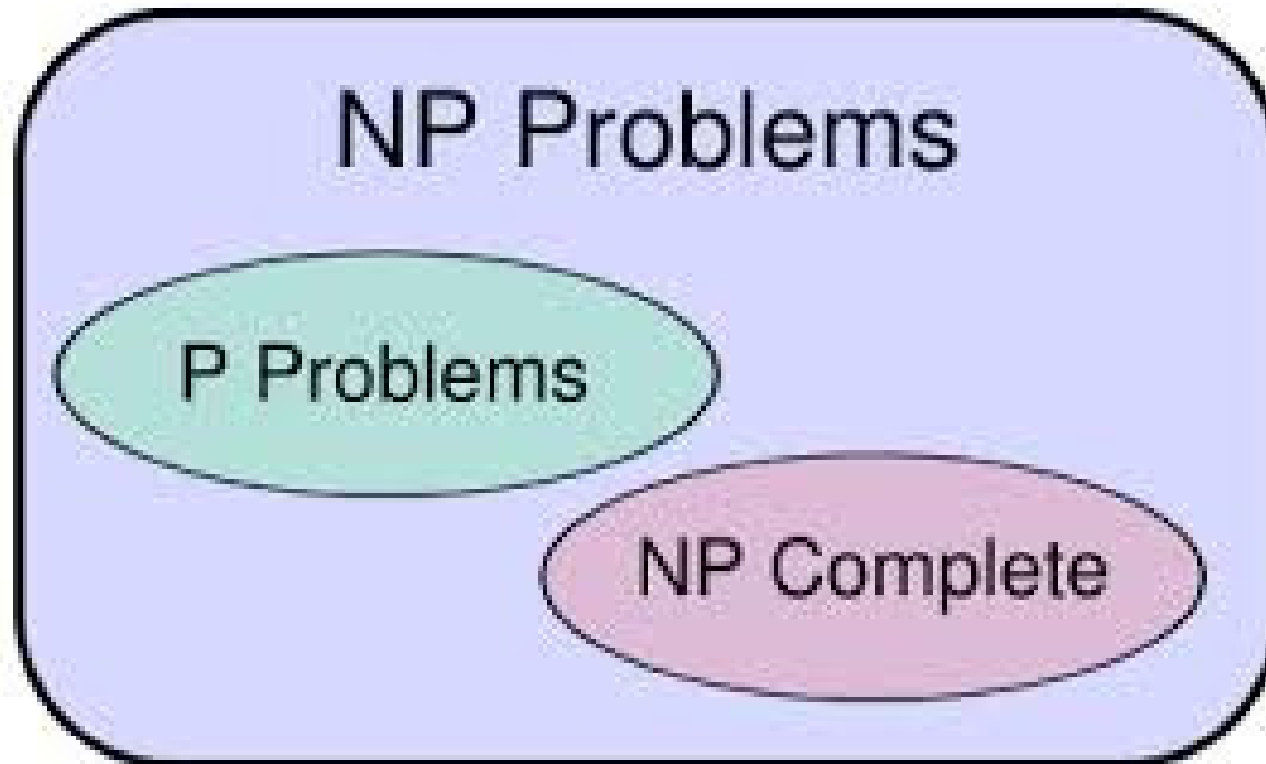


IF5110 Teori Komputasi

Review Teori P dan NP

Oleh: Rinaldi Munir

Program Studi Magister Informatika STEI-ITB



Pendahuluan

- Kebutuhan waktu algoritma yang mangkus bervariasi, mulai dari $O(1)$, $O(\log \log n)$, $O(\log n)$, $O(n)$, $O(n^2)$, dan $O(n^3)$.
- Semua algoritma tersebut digolongkan “bagus” dan dikenal sebagai **solusi polinomial**. Hal ini karena kebutuhannya secara asimptotik dibatasi oleh fungsi polinomial. Misalnya $\log(n) < n$ untuk semua $n \geq 1$.
- Sebaliknya, ada persoalan yang tidak terdapat solusi waktu polinomial untuk menyelesaikannya, misalnya TSP yang memiliki kompleksitas $O(n!)$. Persoalan semacam itu digolongkan sebagai persoalan “sulit” (*hard problem*).

Definisi

- *Polynomial-time algorithm* adalah algoritma yang kompleksitas waktu kasus-terburuknya dibatasi oleh fungsi polinom dari ukuran masukannya.

$T(n) = O(p(n)) \rightarrow p(n)$ adalah polinom dari n

Contoh: Persoalan *sorting* $\rightarrow T(n) = O(n^2)$, $T(n) = O(n \log n)$

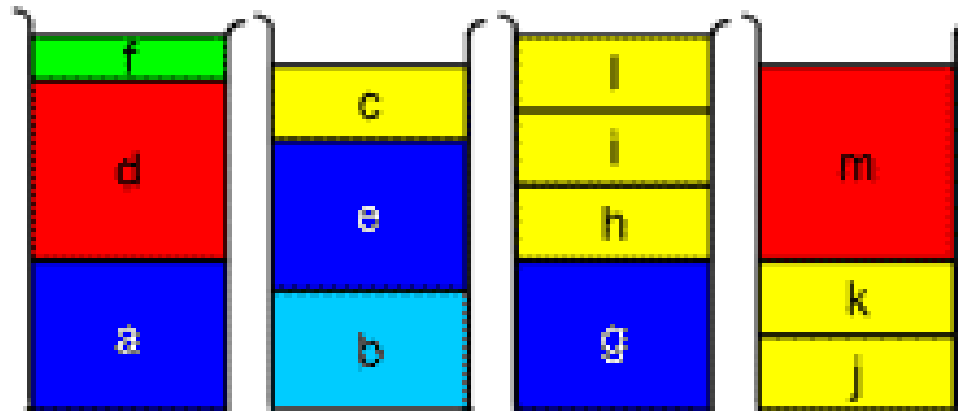
Persoalan *searching* $\rightarrow T(n) = O(n)$, $T(n) = O(\log n)$

Perkalian matriks $\rightarrow T(n) = O(n^3)$, $T(n) = O(n^{2.83})$

- Diluar itu, algoritmanya dinamakan *nonpolynomial-time algorithm*.

Contoh: TSP, *integer knapsack problem*, *graph coloring*,
sirkuit Hamilton, *partition problem*, *bin packing*,
integer linear programming

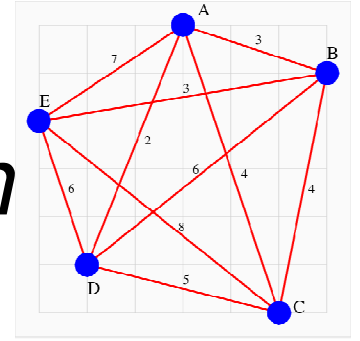
- *Bin-packing problem*: Terdapat sejumlah kardus masing-masing dengan kapasitas C dan n barang berukuran $S_1, S_2, \dots, S_n$. Kemaslah n barang ke dalam M kardus sedemikian sehingga ukuran total barang di dalam setiap kardus tidak melebihi C . Temukan minimal M yaitu paling sedikit jumlah kardus untuk menampung n barang.



Teori NP

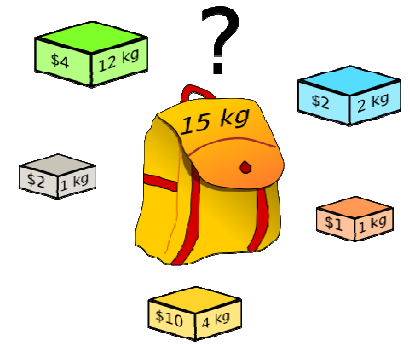
- Dalam membahas teori NP, kita hanya membatasi pada persoalan keputusan (*decision problem*)
- **Persoalan keputusan** adalah persoalan yang solusinya hanya jawaban “yes” atau “no”.
- Setiap persoalan optimasi yang kita kenal memiliki *decision problem* yang bersesuaian.
- Perhatikan beberapa persoalan berikut:

1. Travelling Salesperson Problem



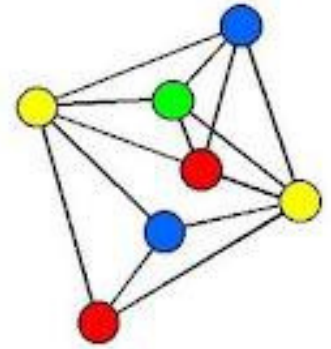
- Diberikan graf berarah dengan bobot (*weight*) pada setiap sisinya. Sebuah tur di dalam graf tersebut dimulai dari sebuah simpul, mengunjungi simpul lainnya tepat sekali dan kembali lagi ke simpul asalnya.
- *Travelling Salesperson Optimization Problem* (TSOP) adalah persoalan menentukan tur dengan total bobot sisi minimal → TSP yang sudah biasa dikenal.
- *Travelling Salesperson Decision Problem* (TSDP) adalah persoalan untuk menentukan apakah terdapat tur dengan total bobot sisinya tidak melebihi nilai d .

2. Knapsack Problem



- Diberikan n buah objek dan sebuah *knapsack* dengan kapasitas W . Setiap objek memiliki profit masing-masing.
- *Integer Knapsack Optimization Problem* adalah menentukan objek-objek yang dimasukkan ke dalam *knapsack* namun tidak melebihi W sehingga memberikan total profit maksimum. → *Knapsack problem yang sudah kita kenal*
- *Integer Knapsack Decision Problem* adalah persoalan untuk menentukan apakah dapat memasukkan objek-objek ke dalam *knapsack* namun tidak melebihi W tetapi total profitnya paling sedikit sebesar P .

3. Graph Colouring Problem



- *Graph-Colouring Optimization Problem* adalah menentukan jumlah minimal warna yang dibutuhkan untuk mewarnai graf sehingga dua simpul bertetangga memiliki warna berbeda. → *Graph Colouring problem* yang kita kenal.
- *Graph-Colouring Decision Problem* adalah menentukan, untuk suatu integer m , apakah terdapat pewarnaan yang menggunakan paling banyak m warna sedemikian sehingga dua simpul bertetangga memiliki warna berbeda.

- Kita belum menemukan algoritma polinomial untuk persoalan optimasi atau persoalan keputusan pada contoh-contoh di atas.
- Namun, jika kita dapat menemukan algoritma polinomial untuk jenis persoalan optimasi tersebut, maka kita juga mempunyai algoritma waktu-polinom untuk persoalan keputusan yang bersesuaian.
- Hal ini karena solusi persoalan optimasi menghasilkan solusi persoalan keputusan yang bersesuaian.

- Contoh: jika pada persoalan *Travelling Salesperson Optimization Problem* (TSOP) tur minimal adalah 120,
- maka jawaban untuk persoalan *Travelling Salesperson Decision Problem* (TSDP) adalah “yes” jika $d \geq 120$, dan “no” jika $d < 120$.
- Begitu juga pada persoalan *Integer Knapsack Optimization Problem*, jika keuntungan optimalnya adalah 230, jawaban untuk persoalan keputusan *integer knapsack* yang berkoresponden adalah “yes” jika $P \leq 230$, dan “no” jika $P > 230$.

P Problems

- ***P Problems*** adalah himpunan semua persoalan keputusan yang dapat dipecahkan oleh algoritma dengan kebutuhan waktu polinom.
- Semua persoalan keputusan yang dapat diselesaikan dalam waktu polinom adalah anggota himpunan *P*.
Contoh: Persoalan mencari apakah sebuah *key* terdapat di dalam sebuah larik adalah *P Problems*.
- Semua persoalan keputusan yang telah ditemukan algoritma dalam waktu polinom termasuk ke dalam *P Problems*.

- Apakah *Travelling Salesperson Decision Problem* (TSDP) termasuk *P Problems*?
- Meskipun belum ada orang yang menemukan algoritma TSDP dalam waktu polinom, namun tidak seorang pun dapat membuktikan bahwa TSDP tidak dapat dipecahkan dalam waktu polinom.
- Ini berarti TSDP *mungkin* dapat dimasukkan ke dalam *P Problems*.

- **Algoritma deterministik** adalah algoritma yang dapat ditentukan dengan pasti apa saja yang akan dikerjakan selanjutnya oleh algoritma tersebut.
- Algoritma deterministik bekerja sesuai dengan *nature* komputer saat ini yang hanya mengeksekusi satu operasi setiap waktu, diikuti oleh operasi selanjutnya (*sequential*), begitu seterusnya.
- **Algoritma non-deterministik** adalah algoritma yang berhadapan dengan beberapa opsi pilihan, dan algoritma memiliki kemampuan untuk menerka opsi pilihan yang tepat.
- Algoritma non-deterministik dijalankan oleh komputer non-deterministik (komputer hipotetik). Komputer non-deterministik memiliki kemampuan mengevaluasi sejumlah tak berhingga operasi secara paralel.

- **Contoh 1: TSDP**

Algoritma non-deterministik TSDP:

for $i := 1$ **to** n **do**

pilih sebuah sisi (*edge*) dari graf G

hapus sisi tersebut dari himpunan sisi

end

- Solusi dapat diverifikasi dengan menghitung semua bobot sisi yang terpilih dan memeriksa apakah jumlahnya lebih kecil dari d
- Memilih sebuah sisi dari graf $\rightarrow O(1)$
- Kompleksitas memilih seluruhnya $\rightarrow O(n)$
- Kompleksitas memverifikasi solusi $\rightarrow O(n)$
- Kompleksitas TSDP total = $O(n) + O(n) = O(n)$

- **Contoh 2 (Persoalan sirkuit Hamilton):** Diberikan sebuah graf G . Apakah G mengandung sirkuit Hamilton? Sirkuit Hamilton adalah sirkuit yang melalui setiap simpul di dalam graf tepat satu

1. Terkalah permutasi semua simpul
2. Periksa (verifikasi) apakah permutasi tersebut membentuk sirkuit. Jika ya, maka itulah solusinya. STOP.

- Ada dua tahap di dalam algoritma non-deterministik:

1. **Tahap menerka (non-deterministik):** Diberikan *instance* persoalan, tahap ini secara sederhana (misalnya) menghasilkan beberapa string *S*. String ini dapat dianggap sebagai sebuah terkaan pada sebuah solusi. String yang dihasilkan bisa saja tidak bermakna (*non-sense*).

2. **Tahap verifikasi (deterministik):** memeriksa apakah *S* menyatakan solusi. Keluaran tahap ini adalah “true” jika *S* merupakan solusi, atau “false” jika bukan.

Contoh pada TSDP:

Tahap verifikasi

```
function verify(G:graph; d:number, S:claimed_tour)
```

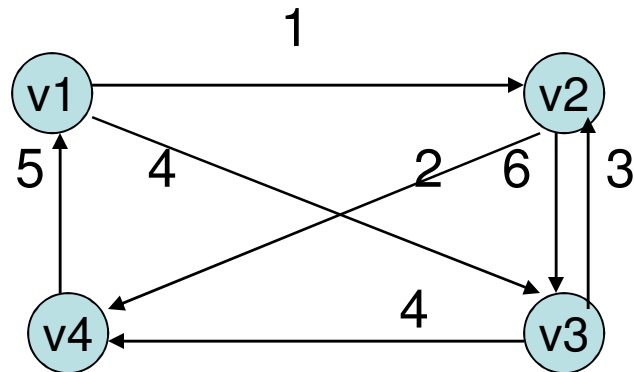
Algoritma

```
    if S adalah tur dan total bobot  $\leq$  d then  
        return true  
    else  
        return false  
    endif
```

- Tahap verifikasi ini dikerjakan dalam waktu polinom, sebab fungsi *verify* membutuhkan waktu $O(n)$, yang dalam hal ini n = jumlah simpul

- Meskipun kita tidak pernah menggunakan algoritma non-deterministik untuk menyelesaikan persoalan, namun kita menyatakan bahwa algoritma non-deterministik “menyelesaikan” persoalan keputusan apabila:
 1. Untuk suatu *instance* dimana jawabanya adalah “yes”, terdapat beberapa string *S* yang pada tahap verifikasi menghasilkan “true”
 2. Untuk suatu *instance* dimana jawabannya adalah “no”, tidak terdapat string *S* yang pada tahap verifikasi menghasilkan “true”.

- Contoh untuk TSDP dengan $d = 15$:



S	Keluaran	Alasan
[v1, v2, v3, v4, v1]	False	Total bobot > 15
[v1, v4, v2, v3, v1]	False	S bukan sebuah tur
^%@12*&a%	False	S bukan sebuah tur
[v1, v3, v2, v4, v1]	True	S sebuah tur, total bobot ≤ 15

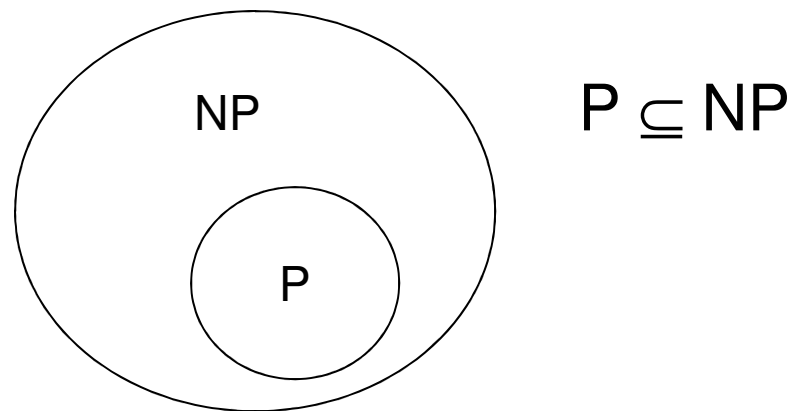
- Kita dapat menyatakan bahwa algoritma non-deterministik “menyelesaikan” TSDP dalam dua tahap tersebut



NP Problems

- *NP = non-deterministik polynomial*
- *Polynomial-time non-deterministic algorithm* adalah algoritma non-deterministik dimana tahap verifikasiya adalah algoritma dengan kebutuhan waktu polinom.
- *NP Problems* adalah himpunan persoalan keputusan yang dapat diselesaikan oleh algoritma non-deterministik dalam waktu polinom.
- Kebanyakan persoalan keputusan adalah NP

- TSDP adalah contoh persoalan NP, sebab jika diberikan sebuah terkaan string (tur), maka dibutuhkan $O(n)$ untuk memverifikasi solusi.
- *Integer Knapsack Decision problem* dan *Graph Coloring Decision Problem* semuanya adalah NP.
- Semua persoalan P juga adalah NP, sebab tahap menerka tidak terdapat di dalam persoalan P. Karena itu, P adalah himpunan bagian dari NP.



- Adakah persoalan yang bukan NP? Ada, yaitu *halting problem*.
- Kita telah menyebutkan sebelumnya bahwa TSDP belum terbukti apakah P atau bukan.
- Dengan kata lain, tidak seorangpun pernah membuktikan bahwa $P \neq NP$ atau $P = NP$.

- Pertanyaan apakah $P = NP$ adalah salah satu pertanyaan penting dalam ilmu komputer.
- Pertanyaan ini penting sebab, seperti telah disebutkan sebelumnya, kebanyakan persoalan keputusan adalah NP.
- Karena itu, jika $P = NP$, maka betapa banyak persoalan keputusan yang dapat dipecahkan secara mangkus dengan algoritma yang kebutuhan waktunya polinom..

- Namun kenyataannya, banyak ahli yang telah gagal menemukan algoritma waktu-polinom untuk persoalan NP.
- Karena itu, cukup aman kalau kita menduga-duga bahwa $P \neq NP$

- The **P versus NP problem** is a major [unsolved problem in computer science](#). Informally, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. It was introduced in 1971 by [Stephen Cook](#) in his seminal paper "The complexity of theorem proving procedures"^[2] and is considered by many to be the most important open problem in the field.^[3] It is one of the seven [Millennium Prize Problems](#) selected by the [Clay Mathematics Institute](#) to carry a US\$1,000,000 prize for the first correct solution.

(Sumber: Wikipedia)

The \$1M question

The Clay Mathematics Institute Millenium Prize Problems:

1. Birch and Swinnerton-Dyer Conjecture
2. Hodge Conjecture
3. Navier-Stokes Equations
4. **P vs NP**
5. Poincaré Conjecture
6. Riemann Hypothesis
7. Yang-Mills Theory