



# Greedy

Tim Olimpiade Komputer Indonesia

# Pendahuluan

Melalui dokumen ini, kalian akan:

- Memahami konsep *greedy*.
- Menyelesaikan beberapa contoh persoalan *greedy* sederhana.



# Greedy

**Greedy** merupakan sebuah teknik dalam strategi penyelesaian masalah, bukan suatu algoritma khusus.



# Konsep Greedy

Suatu persoalan dapat diselesaikan dengan teknik *greedy* jika persoalan tersebut memiliki sifat berikut:

- Solusi optimal dari persoalan dapat ditentukan dari solusi optimal subpersoalan tersebut.
- Pada setiap subpersoalan, ada suatu langkah yang bisa dilakukan yang mana langkah tersebut menghasilkan solusi optimal pada subpersoalan tersebut. Langkah ini disebut juga **greedy choice**.



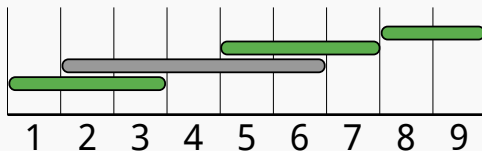
## Contoh Soal: Activity Selection

- Diberikan  $N$  buah aktivitas.
- Aktivitas ke- $i$  dinyatakan dalam  $(a_i.start, a_i.end)$ .
- Artinya, aktivitas ini dimulai pada waktu  $a_i.start$  dan berakhir pada waktu  $a_i.end$ .
- Pada setiap satuan waktu, Anda dapat mengikuti paling banyak satu aktivitas.
- Anda ingin mengatur jadwal sedemikian sehingga Anda bisa ikut aktivitas sebanyak mungkin.



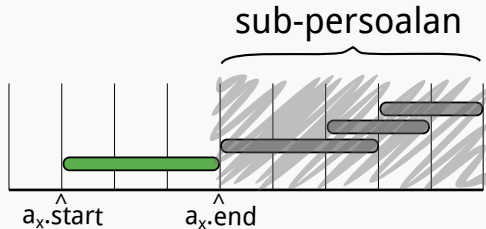
## Contoh Activity Selection

- Sebagai contoh, diberikan 4 buah aktivitas:  
 $[(1, 3), (2, 6), (5, 7), (8, 9)]$ .
- Anda dapat hadir di 3 aktivitas berbeda yang tidak saling tindih, yaitu  $(1, 3)$ ,  $(5, 7)$ , dan  $(8, 9)$ .



## Solusi Activity Selection

- Misalkan kegiatan pertama yang kita ikuti adalah kegiatan ke- $x$ .
- Kegiatan selanjutnya yang diikuti haruslah memiliki waktu awal  $\geq a_x.end$ .
- Lebih jauh lagi, ternyata kita mendapat persoalan yang serupa, hanya saja ukurannya lebih kecil.
- Dengan kata lain, kita memperoleh subpersoalan.



## Solusi Activity Selection (lanj.)

Pertanyaan: aktivitas mana yg akan pertama kali dipilih?

Perhatikan pilihan berikut:

- Memilih aktivitas dengan waktu mulai paling awal.
- Memilih aktivitas dengan durasi paling singkat.
- Memilih aktivitas dengan waktu akhir paling awal.

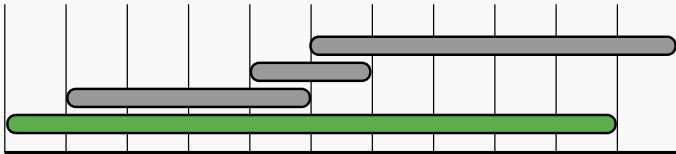




# Memilih Aktivitas Pertama

Memilih aktivitas dengan waktu mulai paling awal:

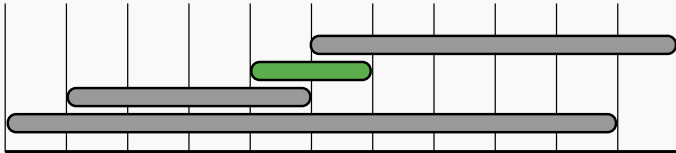
- Bisa jadi ada aktivitas yang mulai lebih awal, tetapi memiliki durasi yang sangat panjang sehingga menyita waktu.
- Memilih aktivitas yang mulai paling awal **belum pasti** optimal.



## Memilih Aktivitas Pertama (lanj.)

Memilih aktivitas dengan durasi paling singkat:

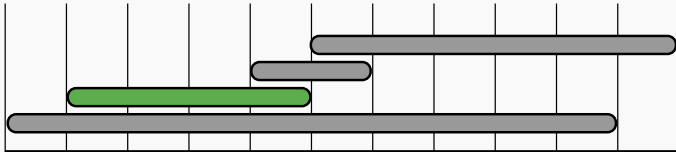
- Bisa jadi aktivitas dengan durasi paling singkat ini memotong dua aktivitas lain yang sebenarnya dapat kita ikuti.
- Pilihan ini juga **belum pasti** menghasilkan solusi optimal.



## Memilih Aktivitas Pertama (lanj.)

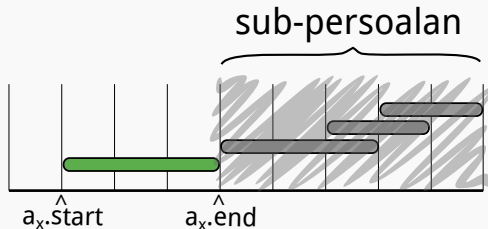
Memilih aktivitas dengan waktu akhir paling awal:

- Dengan memilih aktivitas yang selesai lebih awal, kita mempunyai sisa waktu lebih banyak untuk aktivitas lainnya.
- Tanpa peduli kapan aktivitas ini mulai atau berapa durasinya, memilih yang selesai lebih awal **pasti menguntungkan**.
- Pilihan ini adalah merupakan *greedy choice*, yang **selalu** menghasilkan solusi optimal.



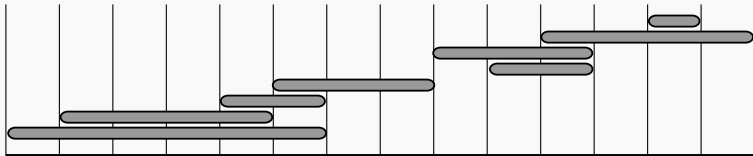
## Penyelesaian Activity Selection

- Kini kita dapat menentukan aktivitas yang akan diikuti pertama kali.
- Selanjutnya kita mendapatkan subpersoalan, yang ternyata dapat diselesaikan dengan cara serupa!



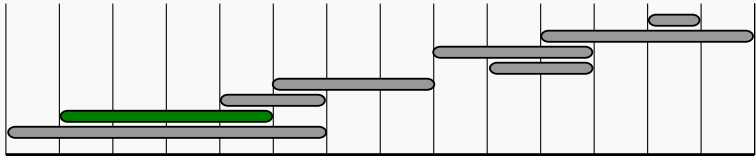
## Contoh Eksekusi Activity Selection

Berikut contoh cara pemilihan aktivitas yang optimal.



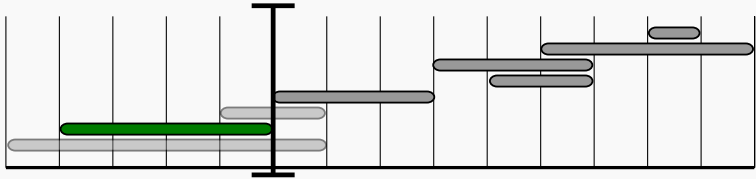
## Contoh Eksekusi Activity Selection (lanj.)

Dimulai dari memilih aktivitas pertama.



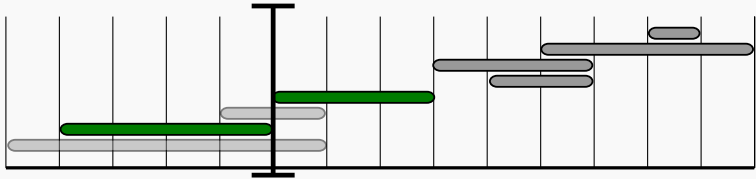
## Contoh Eksekusi Activity Selection (lanj.)

Selanjutnya kita mendapatkan subpersoalan.  
Beberapa aktivitas kini tidak dapat dipilih lagi.



## Contoh Eksekusi Activity Selection (lanj.)

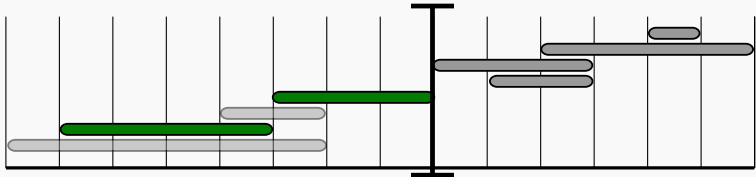
Masalah yang kita hadapi serupa dengan masalah sebelumnya. Kita tinggal memilih aktivitas yang berakhir paling awal.





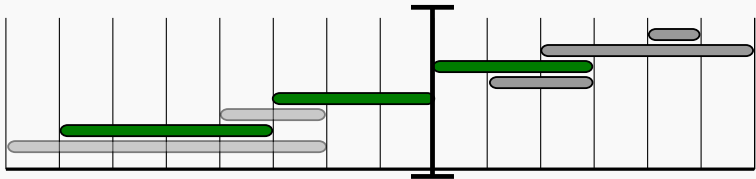
## Contoh Eksekusi Activity Selection (lanj.)

Kembali kita mendapatkan subpersoalan....



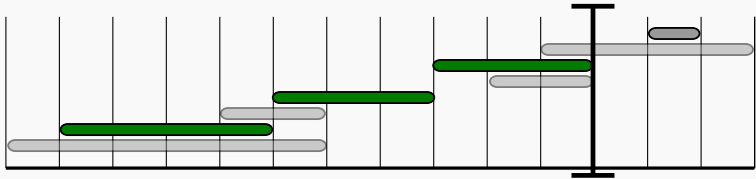
## Contoh Eksekusi Activity Selection (lanj.)

Pilih lagi aktivitas yang berakhir paling awal.



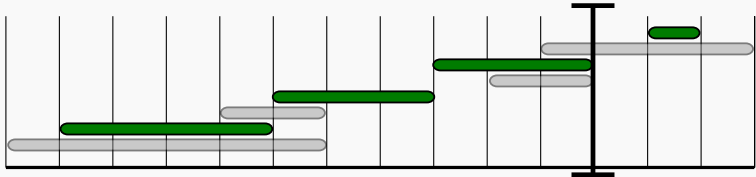
## Contoh Eksekusi Activity Selection (lanj.)

Didapatkan lagi subpersoalan....



## Contoh Eksekusi Activity Selection (lanj.)

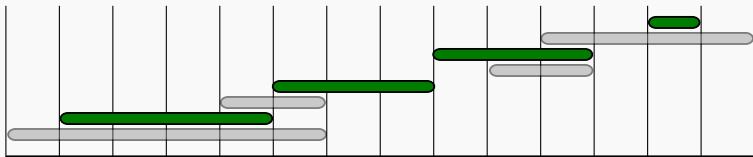
Pilih lagi aktivitas yang berakhir paling awal.



## Contoh Eksekusi Activity Selection (lanj.)

Selesai!

Tidak ada cara lain yang memberikan hasil lebih optimal.



# Implementasi Solusi Activity Selection

```
SOLVEACTIVITYSELECTION( $a[]$ ,  $N$ )  
1  // Urutkan  $a$  secara menaik berdasarkan  $a[i].end$   
2  SORTBYENDINGTIME( $a$ ,  $N$ )  
  
3   $selectedCount = 0$   
4   $startTime = 1$   
5  for  $i = 1$  to  $N$   
6      if ( $a[i].start \geq startTime$ )  
7           $selectedCount = selectedCount + 1$   
8           $startTime = a[i].end + 1$   
9  return  $selectedCount$ 
```



# Analisis Kompleksitas

- Mengurutkan aktivitas berdasarkan waktu berakhirnya dapat dilakukan dalam  $O(N \log N)$ .
- Setelah diurutkan, pemilihan aktivitas dapat dilakukan dalam  $O(N)$ .
- Kompleksitas akhirnya  $O(N \log N)$ .
- Cepat dan efisien!



# Selingan

- *Greedy choice* memungkinkan kita untuk memilih suatu keputusan yang dijamin akan menghasilkan solusi optimal, tanpa peduli ke depannya seperti apa.
- Hal ini memberi kesan "rakus", yaitu hanya mementingkan masalah yang sedang dihadapi dan selalu mengambil keputusan terbaik saat ini.
- Inilah sebabnya teknik ini dinamakan *greedy*.





# Permasalahan pada Algoritma Greedy

Perhatikan contoh soal berikut:

- Anda ingin menukar uang Rp12.000 dengan lembaran uang kertas Rp5.000, Rp2.000, dan Rp1.000.
- Anda ingin menukar dengan jumlah lembaran sesedikit mungkin.



# Permasalahan pada Algoritma Greedy

- *Greedy choice* yang terpikirkan adalah dengan memilih lembar dengan nominal terbesar yang mungkin untuk tiap subpersoalan.
- Pertama kita pilih lembar 5000, sehingga tersisa 7000 lagi yang harus dipecah.
- Selanjutnya kita pilih 5000 lagi dan menyisakan 2000 untuk dipecah.
- Akhirnya, kita pilih 2000 sebagai pecahan terakhir.
- Solusi dari kasus ini adalah dengan menggunakan 3 lembar.



## Permasalahan pada Algoritma Greedy (lanj.)

Dengan soal yang sama, bagaimana jika lembaran rupiah yang beredar bernilai Rp5.000, Rp4.000, dan Rp1.000?



## Permasalahan pada Algoritma Greedy (lanj.)

- Dengan algoritma *greedy*, kita akan menukar 12000 dengan lembaran 5000, 5000, 1000, dan 1000.
- Padahal ada solusi yang lebih baik, yaitu menggunakan 3 lembaran 4000.
- Pada kasus tersebut, *greedy choice* yang tidak selalu dapat menghasilkan solusi optimal.
- Permasalahan ini tidak dapat diselesaikan oleh algoritma *greedy*.
- (Permasalahan ini bisa diselesaikan dengan algoritma *dynamic programming*, yang akan dibahas pada materi berikutnya.)



## Permasalahan pada Algoritma Greedy (lanj.)

- Pembuktian kebenaran algoritma *greedy* tidaklah mudah.
- Biasanya akan ada beberapa pilihan *greedy choice* yang ada, yang mana tidak semuanya bisa menghasilkan solusi optimal.
- Ketika menemukan suatu *greedy choice*, sangat dianjurkan untuk menguji kebenaran dari pilihan tersebut sebelum diimplementasikan.



## Permasalahan pada Algoritma Greedy (lanj.)

- Pengujian yang dapat dilakukan adalah dengan mencoba membuat contoh kasus yang dapat menggagalkan *greedy choice* tersebut.
- Teknik ini biasa disebut *proof by counterexample*.
- Jika ditemukan satu saja contoh kasus yang mana *greedy choice* yang diajukan tidak menghasilkan solusi optimal, maka *greedy choice* tersebut dinyatakan salah.
- Namun, bila Anda tidak bisa menemukan *counterexample*, **belum tentu** algoritma Anda benar.



## Saran

- Algoritma *greedy* terkadang mudah untuk dipikirkan dan mudah untuk diimplementasikan, namun sulit untuk dibuktikan kebenarannya.
- Pembuktian kebenaran algoritma *greedy* bisa jadi membutuhkan pembuktian matematis yang kompleks dan memakan waktu.
- Pada suasana kompetisi, intuisi dan pengalaman sangat membantu untuk menyelesaikan soal bertipe *greedy*.
- Berhati-hati dan teliti saat mengerjakan soal bertipe *greedy*. Perhatikan setiap detail yang ada, karena bisa berakibat fatal.



# Penutup

- Untuk dapat menguasai *greedy*, Anda perlu banyak berlatih dan berpikir secara cerdas.
- Selamat berlatih untuk mengasah "kerakusan" Anda :)

