Desain dan Analisis Algoritma

Semester Genap 2023/2024

PR 4 – Greedy Algorithms, Backtracking, & BnB

Deadline: Rabu, 15 Mei 2024

Petunjuk Pengerjaan:

- Berkas PR Anda harus dibuat dengan cara ditulis tangan di kertas A4 lalu difoto/di scan dan disimpan sebagai satu berkas PDF (bukan di-*zip*). Mengumpulkan selain tipe file PDF dikenakan penalti sebesar 5 poin.
- Format penamaan berkas PR1 NPM Nama.pdf.

  Contoh: PR1 2106123456 John Doe.pdf.
  Penamaan berkas yang tidak sesuai dikenakan penalti sebesar 5 poin.

- Tuliskan Nama dan NPM Anda pada bagian kiri atas setiap halaman pada PR Anda.
- Awali berkas PR Anda dengan pernyataan "Dengan ini saya menyatakan bahwa PR ini adalah hasil pekerjaan saya sendiri" disertai tanda tangan pada halaman pertama berkas PR anda. Tanpa pernyataan ini, PR Anda tidak akan diperiksa.
- Jika ada, tuliskan Nama Kolaborator pada berkas PR Anda. Perhatikan bahwa walaupun Anda sudah menuliskan nama kolaborator, bukan berarti jawaban Anda boleh sama persis dengan kolab- orator Anda. PR adalah tugas individu, bukan tugas kelompok. Pastikan kolaborasi hanya pada sebatas ide pengerjaan, bukan ketika menulis jawaban.
- Anda harus memberikan penjelasan jawaban pada setiap soal. Bila kurang penjelasan, maka akan dikenakan penalti.
- Anda diperbolehkan menghitung menggunakan kalkulator, namun langkah pengerjaan harus dije- laskan. Tidak boleh menulis nilai akhir saja.
- Pelanggaran peraturan kejujuran akademis akan diproses sesuai peraturan yang sudah dijelaskan di BRP.
- Keterlambatan mengumpulkan PR akan dikenai penalti sebesar 30% dari nilai total selama dikumpulkan maksimal satu jam setelah batas pengumpulan PR.

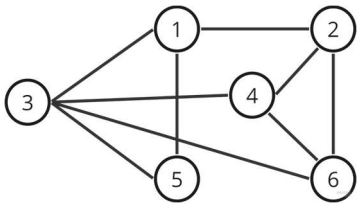
1. Soal 1 (10 + 10 + 10 + 10 = 40 poin)

Greedy Algorithm

Diberikan algoritma *Edge-Coloring* berikut yang menerima input suatu graf yang direpresentasikan dengan array *edges*List berisikan semua *edge*-nya. Algoritma tersebut mengembalikan array colors yang berisikan pewarnaan setiap *edge* dalam graf tersebut, serta memastikan bahwa setiap *edge* yang bersisian dengan *vertex* yang sama tidak memiliki warna identik.

```
EDGE-COLORING(edgesList)
 1: e ← length of edgesList
 2: colors ← list of size e initialized with zeros
 3: i ← 0
 4: color ← 1
 5: while i < e do
 6:    colors[i] ← color
 7:    flag ← false
 8:    for j from 0 to e − 1 do
 9:       if j ≠ i then
10:          commonVertices ← intersection of edgesList[i] and edgesList[j]
11:          if commonVertices is not empty and colors[i] = colors[j] then
12:             color ← color + 1
13:             flag ← true
14:             break
15:    if not flag then
16:       color ← 1
17:       i ← i + 1
18: return colors
```

a. Diberikan graf sebagai berikut:



Terapkan algoritma *Edge-Coloring* pada graf di atas dengan input *edgeList* = [(1,2), (1,3), (1,5), (2,4), (2,6), (3,4), (3,5), (3,6), (4,6)]. Tentukan hasil pewarnaan setiap sisi. Berapa total warna yang digunakan?

b. Diberikan graf yang sama seperti pada bagian (a), namun urutan input *edgeList* yang diberikan diacak dengan input *edgeList* = [(2, 6), (4, 6), (2, 4), (3, 6), (1, 2), (1, 3), (1, 5), (3, 5), (3, 4)].

Terapkan kembali algoritma *Edge-Coloring*. Apakah total warna yang digunakan dan hasil pewarnaannya berbeda dengan hasil sebelumnya? Jika ya, jelaskan mengapa hal ini bisa terjadi.

c. Jelaskan apa yang dilakukan oleh algoritma *Edge-Coloring*. Mengapa algoritma tersebut *greedy*?

d. Apakah algoritma tersebut selalu memberikan hasil pewarnaan dengan banyak warna minimal? Jelaskan jika ya, dan berikan *counterexample* jika tidak.
(Hint: Apakah urutan pewarnaan *edge* berpengaruh terhadap hasil akhir pewarnaan?)

2. Soal 2 (15 poin)

Backtracking

Diberikan sebuah 0-1 *Knapsack Problem* dengan format *input* dan *output* sebagai berikut:

Input: $X = \{w_i, v_i\}$, dengan $i = 1, 2, 3, \dots, n$, dengan kapasitas maksimum sebesar W

Output: nilai maksimum dari $\sum_{i=1}^{n} v_i x_i$ dengan syarat $\sum_{i=1}^{n} w_i x_i \leq W$ serta $x_i \epsilon \{0, 1\}$

Diberikan sebuah tabel input berupa *weight* dan *value* sebanyak $n = 5$, dengan strategi *heaviest on top*.

i	1	2	3	4	5
w_i	1	7	14	4	6
v_i	3	14	84	12	6
v_i/w_i	3	2	6	3	1

Misalkan *knapsack* tersebut mempunyai kapasitas maksimum W=10. Gunakanlah metode dengan pendekatan *backtracking* untuk menyelesaikan permasalahan 0-1 *Knapsack* tersebut.

3. Soal 3 (15 + 10 + 5 + 15 = 45 poin)

Branch and Bound

a. Misalkan Anda diminta untuk mengerjakan ulang soal permasalahan 0-1 *Knapsack* pada soal nomor 2 dengan metode *branch and bound,* dengan input tabel, kapasitas maksimum, serta strategi yang sama. Gambarkanlah *state space tree* yang dibentuk oleh metode tersebut.

b. Berikan penjelasan tentang *tree* yang dibentuk pada bagian (a). Untuk mendapat *full poin*, penjelasan Anda harus mencakup:
   i. Makna *node* dari tree
   ii. Makna *edge* dari tree
   iii. Output
   iv. Himpunan barang yang diambil (yang menghasilkan poin iii)

v. Jika ada, sebutkan node mana saja yang terjadi proses *backtracking*, dan jelaskan mengapa dilakukan *backtracking* (cukup jelaskan 1 *node* saja). Jika tidak ada, jelaskan mengapa tidak ada!

vi. Jika ada, sebutkan node mana saja yang terjadi proses *pruning*, dan jelaskan mengapa dilakukan *pruning* (cukup jelaskan 1 node saja). Jika tidak ada, jelaskan mengapa tidak ada!

c. Dengan asumsi Anda mengerjakan dengan teliti dan tepat, apakah Anda akan mendapatkan output yang sama seperti dengan sebelumnya? Mengapa? Jelaskan dari segi efisiensi (jumlah *node* yang dikunjungi), mana pendekatan yang lebih baik: pendekatan *backtracking* atau *branch and bound*? Elaborasi jawaban Anda!

d. Anda diminta untuk mengerjakan ulang permasalahan pada nomor 2 menggunakan *branch and bound*, tetapi dengan menggunakan strategi lain, yaitu *best density on top*. Gambarkanlah *state space tree* yang dibentuk, serta tuliskanlah output serta himpunan barang yang diambil!

e. Diantara 2 strategi *branch and bound*, manakah yang paling efektif dari segi node yang dikunjungi: pendekatan *heaviest on top* atau *best density on top*? Elaborasi jawaban Anda!