FAKULTAS ILMU KOMPUTER

CSGE602040 - Struktur Data dan Algoritma
Semester Ganjil - 2022/2023
TP 1
Deadline: Rabu, 5 Oktober 2022, 21.00 WIB

Los Polos Armanos Restaurant Service

Revisi Dokumen

- Edisi 1 (2022-09-21 18.00): Dokumen awal
- Edisi 2 (2022-09-22 22.40): Penambahan batasan *query* D
- Edisi 3 (2022-09-23 13:30):
  - Penambahan batasan *query* C
  - Penambahan keterangan apabila telah terjadi *query* B
- Edisi 4 (2022-09-25 10:15): Perubahan batasan nilai dari V
- Edisi 5 (2022-09-26 10:20) Perbaikan penjelasan output
- Edisi 6 (2022-09-27 11:56) Perubahan batasan nilai dari Q
- Edisi 7 (2022-10-04 17:45) Perubahan Ruang Lapar menjadi keadaan restoran padat

Deskripsi

Selamat datang di Los Polos Armanos! Restoran ini merupakan restoran yang memiliki pelayanan yang terbaik di wilayahnya.

Los Polos memiliki M menu dan menu ke-i ($1 \leq i \leq M$) memiliki harga h_i dan tipe t_i. Tipe-tipe dari makanan ada 3, yaitu Airfood, Seafood, dan Groundfood. Los Polos juga memiliki V koki. Setiap koki memiliki spesialisasi S. Spesialisasinya adalah salah satu dari ketiga jenis makanan tersebut.

Los Polos Memiliki N kursi tersedia. Akan ada Y hari pelayanan dimana pada hari ke-i akan ada P_i pelanggan yang datang. Karena masa pandemi COVID, pada hari ke-i, pelanggan ke-j akan memiliki status kesehatan K_{ij}. Status '+' artinya pelanggan tersebut memiliki penyakit COVID, '-' artinya tidak ada penyakit COVID, dan '?' artinya tidak diketahui. Selain itu pelanggan tersebut juga memiliki uang sebanyak U_{ij}.

Los Polos tentunya sangat ketat dalam hal kesehatan pelanggannya, karenanya akan diadakan scanning untuk setiap pelanggan yang datang. Untuk pelanggan yang berstatus '+' tidak akan diizinkan untuk masuk, pelanggan berstatus '-' akan diizinkan untuk masuk, dan pelanggan berstatus '?' diperlukan Advance Scanning.

Advance scanning dilakukan pada pelanggan P_{ij} apabila K_{ij} = '?' dengan mempertimbangkan status dari pelanggan ke-$(j - R_{ij})$ hingga pelanggan ke-$(j - 1)$ pada hari ke-i. Status pelanggan tersebut setelah advance scanning akan mengikuti aturan berikut:

- Jika jumlah pelanggan dengan status '-' $<$ '+', maka status pelanggan P_{ij} adalah '+'

- Jika jumlah pelanggan dengan status '-' ≥ '+', maka status pelanggan P_{ij} adalah '-'

Jika setelah scanning dan kapasitas kursi dalam restoran telah penuh, maka pelanggan yang tersisa akan **tetap masuk ke restoran**, namun **keadaan restoran menjadi padat** ~~harus menunggu di **Ruang Lapar**~~.

Setelah pelanggan memasuki restoran, pelayanan akan dilakukan. Akan terjadi sebanyak X_i pelayanan pada hari ke-i. Pelayanan dapat terjadi dari salah satu 5 kejadian berikut.

1. **P [**$ID\_PELANGGAN$**] [**$INDEX\_MAKANAN$**]**, pelanggan dengan ID $ID\_PELANGGAN$ akan memesan makanan ke-$INDEX\_MAKANAN$. Makanan ini harus dimasak oleh koki yang paling sedikit melayani pelanggan dan sesuai spesialisasinya. Apabila terdapat koki dengan spesialisasi yang sama memiliki jumlah pelayanan yang sama, maka koki dengan ID lebih kecil yang akan melayani.

2. **L**, pesanan yang sudah dipesan oleh pelanggan akan dilayani dan dimasak, kemudian makanan akan dikirim ke pelanggan. Pesanan yang dilayani harus sesuai urutannya dengan urutan pesanan tersebut. Setelah melayani, jumlah pelayanan yang dilakukan koki ditambah 1.

3. **B [**$ID\_PELANGGAN$**],** pelanggan dengan ID $ID\_PELANGGAN$ harus membayar semua makanan yang telah dipesan. Jika pelanggan tidak mampu membayar, pelanggan ini akan di-*blacklist* untuk hari-hari berikutnya. ~~Setelah melakukan pembayaran, pelanggan yang sedang menunggu di Ruang Lapar dapat masuk sesuai dengan urutan tunggu.~~

4. **C [**Q**],** manajer restoran ingin mengetahui Q koki yang paling sedikit melayani pelanggan. Apabila terdapat koki yang memiliki jumlah pelayanan yang sama, maka urutkan dari Spesialisasi Seafood, kemudian Groundfood, kemudian Airfood. Jika koki tersebut memiliki spesialisasi yang sama, maka urutkan berdasarkan ID yang terkecil.

5. **D [**$COST\_A$**] [**$COST\_G$**] [**$COST\_S$**],** manajer restoran menawarkan penawaran menarik kepada pembeli yang ingin membeli **tepat satu** dari setiap menu yang ada. Selain membeli dengan harga normal, pembeli dapat membentuk Paket A, Paket G, dan Paket S dimana masing-masing paket tersebut harus **merupakan kumpulan makanan dari index i hingga index j (dengan i < j) sesuai urutan makanan. Kumpulan makanan tersebut harus diawali dan diakhiri dengan tipe makanan yang sama dengan tipe paket**. Apabila suatu paket telah terbentuk, harga masing-masing makanan pada paket tersebut mengikuti harga $COST\_A$, $COST\_G$, atau $COST\_S$ sesuai dengan tipenya. Setiap jenis paket hanya dapat dibentuk sekali, sehingga jumlah maksimal paket yang dapat terbentuk adalah 3. Pelanggan perlu mencari harga paling murah dengan menggabungkan harga menu satuan dengan harga paket yang ditawarkan.

Format Masukan

- Baris pertama berisi M, banyaknya menu yang tersedia pada restoran

- M baris berikutnya berisi Harga h_i, dan Tipe t_i dari masing-masing makanan yang dipisahkan dengan spasi.
- Baris berikutnya berisi V, yaitu banyaknya koki yang bekerja di restoran.
- Baris berikutnya berisi V karakter yang merupakan spesialisasi dari masing masing koki yang dipisahkan dengan spasi. Urutan dari koki merupakan ID koki tersebut (dimulai dari angka 1).
- Baris berikutnya berisi P, yaitu banyaknya pelanggan secara keseluruhan.
- Baris berikutnya berisi N, yaitu banyaknya kursi yang tersedia pada restoran.
- Baris berikutnya berisi Y, yaitu jumlah hari restoran beroperasi.
- Selanjutnya akan terdapat Y himpunan masukan dengan spesifikasi sebagai berikut:
    - Baris pertama berisi P_i, yaitu jumlah pelanggan yang datang ke restoran pada hari ke-i.
    - P_i baris berikutnya merupakan antrian pelanggan yang datang ke restoran pada hari ke-i. Setiap barisnya berisi I_{ij}, K_{ij}, dan U_{ij} yang dipisahkan oleh spasi, serta R_{ij} apabila $K_{ij} = \text{'?'}$. Secara berurutan, masing-masing melambangkan ID, status kesehatan, jumlah uang, dan *range* untuk *Advance Scanning*.
    - Baris berikutnya berisi X_i, yaitu jumlah pelayanan yang akan dilakukan pada hari ke-i.
    - X_i baris berikutnya akan berisi pelayanan yang sesuai dengan format pelayanan yang telah dijabarkan sebelumnya.

Format Keluaran

- Setelah melakukan input pelanggan yang ingin makan di hari tertentu, keluarannya berupa
    - 0 apabila pelanggan tersebut berstatus positif
    - 1 apabila pelanggan tidak ada masalah
    - 2 apabila pelanggan masuk dalam keadaan restoran padat ~~dialihkan ke Ruang Lapar~~
    - 3 apabila pelanggan tersebut telah di-*blacklist*

  Urutan pengecekannya dimulai dari di-*blacklist* atau tidak, berstatus positif atau tidak, lalu padat tidaknya restoran ~~dialihkan ke Ruang Lapar atau tidak~~.
- Hasil dari perintah P mengeluarkan ID Koki yang mengambil pesanan.
- Hasil dari perintah L mengeluarkan ID Pelanggan yang memesan.
- Hasil dari perintah B mengeluarkan
    - 0 apabila uang pelanggan tidak mencukupi
    - 1 apabila uang pelanggan mencukupi
- Hasil dari perintah C mengeluarkan ID Koki yang berada pada urutan Q teratas.
- Hasil dari perintah D mengeluarkan harga yang paling minimum untuk dapat membeli seluruh menu makanan.

Batasan

$1 \leq M \leq 50\,000$

$1 \leq h_i \leq 100\,000$

$S, t_i \in \{\text{'A'}, \text{'G'}, \text{'S'}\}$

$1 \leq V \leq 1\,000\,000$

$1 \leq P \leq 100\,000$

$1 \leq N \leq 50\,000$

$1 \leq Y \leq 5$

$1 \leq I_{ij} \leq 100\,000$

$K_{ij} \in$ {'+', '-', '?'}

$1 \leq U_{ij} \leq 100\,000$

$1 \leq R_{ij} < j$

$1 \leq X_i \leq 200\,000$

Batasan *Query* Pelayanan

- Pada *query* P, dijamin terdapat pelanggan dengan ID $ID\_PELANGGAN$ dan makanan dengan index $ID\_MAKANAN$**.**
- Pada *query* C, dijamin $Q \leq V/2$.
- Pada setiap *test case,* jumlah pemanggilan *query* C ≤ 5
- Pada *query* D, dijamin $1 \leq COST\_A, COST\_G, COST\_S \leq 100\,000.$
- Jika terdapat *query* D, dijamin $1 \leq M \leq 1\,000.$
- Pada setiap *test case*, jumlah pemanggilan *query* D $\leq 2\,500$
- Jumlah *query* P dan L dijamin berjumlah sama (setiap pemesanan pasti dilayani).
- Pada setiap waktu, dijamin tidak ada kasus dimana jumlah *query* L $>$ jumlah *query* P (pasti ada pemesanan yang dapat dilayani).
- Semua pelanggan pasti membayar apabila setidaknya telah terjadi sekali pemesanan.
- Pelanggan yang telah membayar ~~tidak dipindahkan ke Ruang Lapar dan~~ tidak akan melakukan pemesanan lagi pada hari tersebut.
- Setidaknya ada satu koki untuk masing-masing spesialisasi A, G, dan S.
- Pelanggan pertama pada setiap hari dijamin tidak memiliki status kesehatan '?'.
- *Range* dari pelanggan berstatus '?' minimal 1 dan maksimal posisi indeks pelanggan (dengan asumsi indeks dimulai dari 0).

Contoh Masukan 1

```
5
10000 A
15000 G
20000 S
5000 G
17000 S
4
G A S S
4
2
1
4
2 + 20000
3 - 13000
```

```
1 ? 30000 2
4 - 20000
12
P 3 3
P 1 5
L
P 3 5
L
L
B 1
P 4 1
B 3
L
B 4
C 2
```

Contoh Keluaran 1

```
0 1 1 2
3
3
3
4
1
3
1
2
0
4
1
1 4
```

Penjelasan Contoh 1

1. Setiap pelanggan yang datang harus melalui *scanning* terlebih dahulu.
    a. Pelanggan ID 2 berstatus positif sehingga tidak diperbolehkan masuk.
    b. Pelanggan ID 3 berstatus negatif sehingga diperbolehkan masuk.
    c. Pelanggan ID 1 tidak diketahui statusnya sehingga perlu dilakukan Advance Scanning terlebih dahulu berdasarkan *range* 2 pelanggan sebelumnya, yaitu pelanggan 2 dan 3. Jumlah pelanggan negatif ada 1 dan jumlah pelanggan positif ada 1. Karena jumlah negatif sama dengan positif, maka Pelanggan 1 menjadi berstatus negatif dan bisa masuk.
    d. Pelanggan ID 4 berstatus negatif sehingga diperbolehkan masuk. Karena kapasitas kursi hanyalah 2, restoran menjadi padat ~~Pelanggan 4 harus masuk ke Ruang Lapar terlebih dahulu sampai salah satu pelanggan melakukan pembayaran~~.

2. Pelanggan 3 memesan Makanan 3 yang merupakan Seafood, sehingga akan dimasak oleh koki dengan tipe S, yaitu antara Koki 3 atau Koki 4. Walaupun sama-sama melayani 0 makanan, koki yang akan melayani adalah Koki 3 karena memiliki ID yang lebih kecil. Program mengeluarkan *output* 3.

3. Pelanggan 1 memesan Makanan 5 yang merupakan Seafood, sehingga akan dimasak oleh koki dengan tipe S yang memiliki pelayanan terendah, yaitu Koki 3 (jumlah pelanggan yang dilayani Koki 3 belum bertambah karena belum melakukan pelayanan L). Program mengeluarkan *output* 3.

4. Pelanggan 3 dilayani. Jumlah pelayanan yang telah dilakukan Koki 3 bertambah. Program mengeluarkan *output* 3.

5. Pelanggan 3 memesan Makanan 5 yang merupakan Seafood. Karena Koki 3 telah melayani 1 makanan, Koki 4 yang masih melayani 0 makanan akan memasak pesanan ini. Program mengeluarkan *output* 4.

6. Pelanggan 1 dilayani. Jumlah pelayanan yang telah dilakukan Koki 3 bertambah. Program mengeluarkan *output* 1.

7. Pelanggan 3 dilayani. Jumlah pelayanan yang telah dilakukan Koki 4 bertambah. Program mengeluarkan *output* 3.

8. Pelanggan 1 membeli 1 makanan, yaitu Makanan 5 seharga 17.000. Karena Pelanggan 1 memiliki uang sebanyak 30.000, maka pembayaran berhasil dilakukan dan program mengeluarkan *output* 1. ~~Karena ada pelanggan yang telah selesai makan, maka Pelanggan 4 yang sedang berada di Ruang Lapar diperbolehkan masuk.~~

9. Pelanggan 4 memesan Makanan 1 yang merupakan Airfood, maka pesanan akan dimasak oleh koki dengan tipe A dengan jumlah pelayanan terendah, yaitu Koki 2. Program mengeluarkan *output* 2.

10. Pelanggan 3 membeli Makanan 3 dan 5 dengan total harga 37.000. Uang yang dimiliki Pelanggan 3 hanya 13.000 sehingga tidak cukup untuk membayar. Pelanggan 3 pun di-*blacklist* dan program mengeluarkan *output* 0.

11. Pelanggan 4 dilayani. Jumlah pelayanan yang telah dilakukan Koki 2 bertambah. Program mengeluarkan *output* 4.

12. Pelanggan 4 membeli Makanan 1 seharga 10.000. Karena Pelanggan 4 memiliki uang sebanyak 20.000, maka pembayaran berhasil dilakukan dan program mengeluarkan *output* 1.

13. Jumlah pelayanan koki di akhir hari setelah diurutkan berdasarkan jumlah pelayanan terkecil: Koki 1 = 0, Koki 4 = 1, Koki 2 = 1, Koki 3 = 2. Koki 4 menempati posisi kedua walaupun Koki 2 juga melayani 1 pelanggan karena prioritas spesialisasi S > G > A. Program mengeluarkan *output* 1 4.

Contoh Masukan 2

```
5
21000 A
15000 G
11000 S
7000 A
30000 S
4
S A G S
4
2
1
1
1 - 20000
2
D 12000 17000 22000
D 15000 21000 25000
```

Contoh Keluaran 2

```
1
78000
84000
```

Penjelasan Contoh 2

Makanan 1	Makanan 2	Makanan 3	Makanan 4	Makanan 5
A (21rb)	G (15rb)	S (11rb)	A (7rb)	S (30rb)

Paket yang dapat terbentuk dengan menu yang tersedia adalah sebagai berikut:

- [A G S A], S
- A, G, [S A S]
- A, G, S, A, S

1. Untuk D 12000 17000 22000, maka

[A G S A], S	4 * COST_A + Harga Makanan ke-5 = 48000 + 30000 = 78000
A, G, [S A S]	Harga Makanan ke-1 + Harga Makanan ke-2 + 3 * COST_S = 21000 + 15000 + 66000 = 102000
A, G, S, A, S	Harga Makanan ke-1 + ….. + Harga makanan ke-5

	= 21000 + 15000 + 11000 + 7000 + 30000 = 84000

Konfigurasi [A G S A], S menghasilkan harga paling murah, sehingga harga paling minimum untuk membeli seluruh makanan adalah 78000.

2. Untuk D 15000 21000 25000, maka

[A G S A], S	4 * COST_A + Harga Makanan ke-5 = 60000 + 30000 = 90000
A, G, [S A S]	Harga Makanan ke-1 + Harga Makanan ke-2 + 3 * COST_S = 21000 + 15000 + 75000 = 111000
A, G, S, A, S	Harga Makanan ke-1 + ….. + Harga makanan ke-5 = 21000 + 15000 + 11000 + 7000 + 30000 = 84000

Konfigurasi A G S A S (Ala Carte / harga satuan semua) menghasilkan harga paling murah, sehingga harga paling minimum untuk membeli seluruh makanan adalah 84000.

Contoh Masukan 3

```
3
10000 A
14000 S
11000 G
4
S A G S
4
2
2
3
1 - 20000
2 - 10000
3 - 15000
3
P 2 2
L
B 2
3
2 - 30000
3 - 25000
4 - 12000
3
```

```
P 3 2
L
B 3
```

Contoh Keluaran 3

```
1 1 2
1
2
0
3 1 1
4
3
1
```

Penjelasan Contoh 3

Hari Pertama

1. Kapasitas restoran hanya 2 sehingga restoran menjadi padat ketika Pelanggan 3 masuk ~~yang dapat masuk hanya Pelanggan 1 dan Pelanggan 2. Pelanggan 3 masuk ke Ruang Lapar~~.
2. Pelanggan 2 memesan Makanan 2 yang merupakan Seafood, sehingga akan dimasak oleh koki dengan tipe S, yaitu antara Koki 1 atau Koki 4. Walaupun sama-sama melayani 0 makanan, koki yang akan melayani adalah Koki 1 karena memiliki ID yang lebih kecil. Program mengeluarkan *output* 1.
3. Pelanggan 2 dilayani. Jumlah pelayanan yang telah dilakukan Koki 1 bertambah. Program mengeluarkan *output* 2.
4. Pelanggan 2 membeli Makanan 2 seharga 14.000. Uang yang dimiliki Pelanggan 2 hanya 10.000 sehingga tidak cukup untuk membayar. Pelanggan 2 pun di-*blacklist* dan program mengeluarkan *output* 0.

Hari Kedua

1. Pelanggan 2 di-*blacklist* sehingga mengeluarkan output 3. Pelanggan lainnya diperbolehkan masuk dan kapasitas restoran masih mencukupi.
2. Pelanggan 3 memesan Makanan 2 yang merupakan Seafood. Karena Koki 1 telah melayani 1 makanan, Koki 4 yang masih melayani 0 makanan akan memasak pesanan ini. Program mengeluarkan *output* 4.
3. Pelanggan 3 dilayani. Jumlah pelayanan yang telah dilakukan Koki 4 bertambah. Program mengeluarkan *output* 3.
4. Pelanggan 3 membeli Makanan 2 seharga 14.000. Karena Pelanggan 4 memiliki uang sebanyak 25.000, maka pembayaran berhasil dilakukan dan program mengeluarkan *output* 1.

Informasi Tambahan Test-case

Deskripsi	Test Case
Mengandung Query P, L, B	1 - 20, 56 - 70
Mengandung Query D saja	21 - 30, 71 - 85
Mengandung Query P, L, B, C	31 - 40, 86 - 95
Mengandung Query P, L, B, C, D	41 - 55, 96 - 100