Harder DP & Greedy

William Gozali
Pelatnas 2 TOKI 2015

DP + cari konfigurasi ke-K

Contoh soal:

- SPOJ – CTPLUCKY
- SPOJ – ALONE
- APIO 2009 – DNA

Contoh Lain

- Buat sebuah string dengan panjang N karakter
- String hanya terdiri dari karakter 'A' dan 'B'
- String tidak boleh mengandung substring “BAB”
- Tentukan string yang memenuhi syarat tersebut yang leksikografis ke-K

Solusi

- Lakukan DP untuk menghitung berapa banyaknya kemungkinan untuk membentuk string-nya
- Misalkan DP kita bekerja dari depan (mengisi karakter pertama, kedua, dst), dan pada rekurensnya mencoba mengisikan 'A' dulu ketimbang 'B'

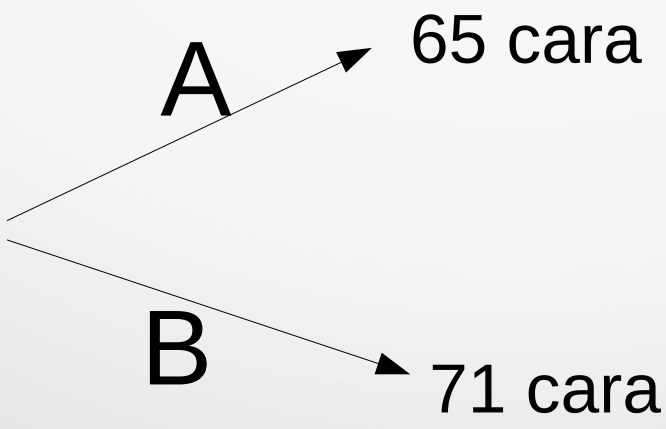
Solusi

- Setelah seluruh tabel DP terisi, kita punya informasi:
  - Jika pada karakter pertama diisi dengan 'A', maka berapa banyak kemungkinan string yang dapat terbentuk
  - Jika pada karakter kedua diisi dengan 'B', maka berapa banyak kemungkinan string yang dapat terbentuk
- Dengan informasi tersebut, kita bisa tahu harus mengisikan 'A' atau 'B' ke karakter pertama
- Lakukan hal yang serupa untuk karakter kedua, ketiga, dst

Solusi (lanj.)

- Misalkan gambar di bawah menyatakan transisi DP untuk karakter pertama

- Untuk K = 50, isikan 'A' dan lanjut cari solusi ke-50
- Untuk K = 127, isikan 'B' dan lanjut cari solusi ke-(127-65)



Solusi (lanj.)

- Kompleksitas akhir: kompleksitas DP + kompleksitas traversal
- Tabel harus disimpan, tidak boleh dibuang seperti pada teknik flying table

DP + struktur data

- Struktur data bisa digunakan untuk membantu komputasi dalam suatu state DP
- Contoh:
  - SPOJ QTGIFT1 (deque/heap)
  - Latihan – Barisan Tentara (range tree)

Contoh lain – Kue (ICPC Jakarta 2014)

- Ada permainan yang diikuti 2 orang (A dan B)
- A dan B sudah diatur giliran jalannya menurut sebuah string sepanjang 2N karakter (hanya berisi 'A 'dan 'B')
- Ada M kue
- Pada setiap giliran, orang itu harus makan kue antara 1..K
- Orang yang tidak bisa menjalankan gilirannya kalah

- 1 <= N, M, K <= 1000

Solusi

$$dp(i,kue)=\begin{cases} 0 & ,kue=0 \\ \max\limits_{1\le t\le min(kue,K)} dp(i-1,kue-t) & ,giliran[i]=A \\ \min\limits_{1\le t\le min(kue,K)} dp(i-1,kue-t) & ,giliran[i]=B \end{cases}$$

- dp(i, kue) = apakah A menang untuk giliran[i..2N] dan masih tersisa “kue” kue
- A berusaha mendapatkan 1 (A menang)
- B berusaha mendapatkan 0 (A kalah)
- Kompleksitas O(NMK)

Solusi (lanj.)

- Percepat dengan segment tree menjadi O(NM log M)
- Buat segment tree untuk setiap baris (untuk i)

	a	b	c	d	e		
						X	

Misalkan K = 5, untuk mengisi sel X,
lakukan query max/min pada segmen [a,e]

Solusi (lanj.)

- Perhatikan bahwa yang kita butuhkan sebenarnya hanya
  - “adakah angka 0 pada segmen ini?”, atau
  - “adakah angka 1 pada segmen ini?”
- Jadi kita bisa membuat tabel counting untuk angka 0 dan angka 1 baris sebelumnya
- Waktu build tabel counting: O(N) per baris
- Waktu query: O(1) per query
- Kompleksitas solusi menjadi O(NM)

Solusi alternatif

- Perhatikan bahwa lebar segmen yang ditanya selalu sebesar K
- Gunakan fixed-segment RMQ
- Info lanjut: http://kupaskode.blogspot.com/2014/03/fixed-size-rmq.html
- Waktu build tabel: O(N) per baris
- Waktu query: O(1) per query
- Kompleksitas solusi menjadi O(NM)

Greedy...

- Butuh banyak observasi
- Coba pikirkan melalui trik-trik berikut:
  - Sort menurut suatu kriteria
  - Bekerja dari belakang
  - Pikirkan bagaimana kriteria solusi optimal, lalu cari cara untuk mencapai itu dengan menyisihkan kemungkinan-kemungkinan yang tidak optimal
  - Amati kasus-kasus yang ada
  - Amati constraint yang mencurigakan
- Kemudian suatu hal yang penting...

ANDA HARUS RAKUS

Contoh: Pindahan

- Ada N barang
- Barang ke-i memiliki berat B[i]
- Untuk setiap pasang barang, barang yang satu pasti kelipatan dari barang yang lain
- Ada M kotak
- Kotak ke-i muat barang-barang dengan total berat <= M[i]
- Tentukan banyaknya barang maksimal yang bisa dimuat ke kotak-kotak

Contoh: Pindahan

- Ada N barang
- Barang ke-i memiliki berat B[i]
- Untuk setiap pasang barang, barang yang satu pasti kelipatan dari barang yang lain
- Ada M kotak
- Kotak ke-i muat barang-barang dengan total berat <= M[i]

Sangat mencurigakan!

Observasi 1

- Urutkan berat barang
- Barang yang diambil pasti K barang teringan pertama
- Karena jika ada solusi optimal yang tidak menggunakan K barang teringan, kita selalu bisa menukarnya dengan menggunakan K barang teringan

Observasi 2

- Setiap barang bisa dinyatakan dalam bentuk perkalian beberapa bilangan
- Jika setiap barang diurutkan, maka bentuknya:
  - a
  - a*b
  - a*b*c
  - a*b*c*d
  - …

Observasi 3

- Jika barang dengan berat B dimasukkan ke kotak dengan kapasitas M, maka bisa dianggap kita kehilangan kotak dengan kapasitas M dan mendapat kotak baru dengan kapasitas M-B

Solusi 1

- Binary search K (banyaknya barang teringan pertama yang diambil)
- Gunakan priority_queue untuk menyimpan kotak-kotak yang tersedia
- Mulai dari barang ke-K teringan, cari kotak terbesar dan masukkan barang ini ke dalamnya
- Ulangi sampai barang ke-1
- Jika selalu ada kotak, naikkan K
- Jika suatu ketika tidak ada kotak yang sesuai lagi, turunkan K
- Kompleksitas $O(N \log^2 N)$

Solusi 2

- Dengan observasi 2, diketahui:
- B[i] >= B[j] + B[k], untuk setiap B[i] > B[j] dan B[i] > B[k]
- Dengan begitu, kita bisa mulai dengan berusaha memasukkan barang ke-N hingga barang ke-X (berjalan mundur) melalui strategi yang sama dengan solusi 1
- Jika suatu ketika barang ke-X tidak bisa dimuat ke kotak manapun, **keluarkan** barang ke-N dari kotaknya
- Maknanya sama dengan kita mendapatkan kotak baru dengan kapasitas B[N]. Masukkan sebanyak mungkin barang ke-X sampai barang ke-(X-p) ke dalam kotak ini, dan lanjutkan jalannya algoritma

Solusi 2 (lanj.)

- Berhenti ketika barang ke-1 sampai barang ke-K sudah dimasukkan. K adalah jawabannya
- Kompleksitas O(N log N)