Analisi della riducibilità mutua tra versioni del primo esempio di PD su linea (`prima_PD_su_linea_interactive`)

Riproponiamo il problema `prima_DP_su_linea`:

> di n oggetti in fila indiana, ognuno con un suo valore, scegli quali prendere volendo massimizzare il valore totale degli oggetti presi ma senza prendere due oggetti adiacenti (consecutivi nella fila) o che sono adiacenti ad uno stesso oggetto.

per il quale avevamo già proposto la seguente formulazione:

> **Formulazione:** E' dato un vettore A di n celle con indici in $[0, n) := \{0, 1, ..., n-1\}$. Ogni sua cella $A_i, i \in [0, n)$, contiene un intero nell'intervallo $[0, 100)$. Tra i sottoinsiemi S di $[0, n)$ per cui $i, j \in S$ implichi $|j - i| \geq 3$ dovete produrne uno che massimizzi $\sum_{i \in S} A_i$.

I seguenti quattro goal si prestano ad una versione interattiva del problema dove tu non hai accesso diretto al vettore A ma ad un oracolo che, su sotto-vettori A' di A qualsiasi, sà risolvere versioni solo parziali della questione.

- **`search2opt` (ridurre la versione di search a quella di ottimizzazione)** dato un oracolo che torna il valore ottimo per un qualsiasi sotto-vettore A' di A, sapresti ricostruire una soluzione ottima per l'istanza originale A? Minimizza il numero di chiamate poste all'oracolo.
- **`opt2dec` (ridurre la versione di ottimizzazione a quella di decisione)** calcola il valore ottimo per l'istanza originale A avvalendoti di un oracolo che, su generica coppia (A', T') con A' sotto-vettore di A e T' intero, dica se il valore ottimo per A' è almeno T'. Minimizza il numero di chiamate poste all'oracolo.
- **`search2dec` (ridurre la versione di search a quella di decisione)** produci una soluzione ottima per l'istanza originale A disponendo di un oracolo che, su generica coppia (A', T') con A' sotto-vettore di A e T' intero, dica se il valore ottimo per A' è almeno T'. Minimizza il numero di chiamate poste all'oracolo.
- **`opt2opt` (ridurre la versione di ottimizzazione a sè stessa)** assumi però che l'oracolo sia disponibile a risolverti solo istanze in cui i sotto-vettori A' che gli sottoponi abbiano tutti meno di n celle. Altrimenti è troppo facile, no?

Per produrre all'oracolo il sotto-vettore $A' = A[i, j] := A_i, A_{i+1}, ..., A_j$ ti basta fornirgli la coppia di interi i, j con $0 \leq i \leq j < n$ (per esibire la competenza `opt2opt` attieniti a domande con $j - i < n$).

Protocollo di comunicazione

Il tuo programma interagirà col server leggendo dal proprio canale `stdin` e scrivendo sul proprio canale `stdout`. La prima riga di `stdin` contiene T, il numero di testcase da affrontare. All'inizio di ciascun testcase, trovi sulla prossima riga di `stdin` uno dei quattro possibili tag di goal: `search2opt`, `opt2dec`, `search2dec`, `opt2opt`, seguito da uno spazio e dal numero n (la lunghezza del vettore A). Inizia ora un loop per uscire dal quale devi stampare su `stdout` una riga che inizia col punto esclamativo e contiene la tua risposta definitiva (stampa anche solo "`!`" per eventuali goal che non hai ancora implementato correttamente, così da poter essere valutato sugli altri). Il punto esclamativo va seguito da un singolo numero nei goal `opt2dec` e `opt2opt`, o dalla sequenza degli indici degli oggetti da prendere nei goal `search2opt`, `opt2dec`, `search2dec`. Scrivi invece una riga della forma "`?i j`" con $i, j \in \mathbb{N}, 0 \leq i \leq j < n$ per porre una domanda all'oracolo previsto per quel tipo di goal (ossia all'oracolo per la versione

decisione del problema per i goal `opt2dec` e `search2dec`, oppure ad un oracolo per la forma di ottimizzazione del problema per i goal `search2opt` e `opt2opt`). Ad immediato seguire dell'aver così posto la tua domanda, leggi la risposta dell'oracolo da `stdin` (il primo oracolo risponde in 0/1 mentre il secondo risponde con numeri naturali qualsiasi).

Nota 1: Ogni tua comunicazione verso il server deve essere collocata su una diversa riga di `stdout` e ricordati di forzarne l'invio immediato al server effettuando un flush del tuo output!

Nota 2: Il tuo dialogo diretto col server (ossia ancora prima di aver scritto una sola riga di codice) può aiutarti ad acquisire il corretto protocollo di comunicazione tra il server e il tuo programma. Verrai così anche a meglio conoscere il problema e sarai più pronto a progettare come condurre la fase di codifica. Il file `results.txt` verrà comunque scaricato nel folder `output` alla fine dell'interazione col server se la termini con `Ctrl-D`.

Esempio

Le righe che iniziano con '?' o '!' sono quelle inviate dallo studente o da suo programma, le altre sono quelle inviate dal server. Inoltre: di ciascuna riga è di mero commento quella parte che comincia col primo carattere di cancelletto '#'.

```
5      # numero di testcase/istanze/partite
opt2dec 10  #   il server comunica il setting del primo testcase (n=10)
...     # il problem solver ...
...
```

Subtask

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 1 secondo.

I testcase sono raggruppati nei seguenti subtask.

1. **[48 pts← 4 istanze da 3 + 3 + 3 + 3 punti] `tiny`:** $N \leq 10$
2. **[36 pts← 3 istanze da 3 + 3 + 3 + 3 punti] `small`:** $N \leq 50$
3. **[24 pts← 2 istanze da 3 + 3 + 3 + 3 punti] `medium`:** $N \leq 100$

Nella descrizione dei subtask è specificato quanti punti si acquisiscono su ciascuna istanza di quel subtask.

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL>  connect -x <token> -a size=medium
prima_PD_su_linea_interactive  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

`tiny`, `small`, `medium`.

Il valore di default per l'argomento `size` è `medium` che include tutti i testcase.

[1] <URL> server esame: `wss://ta.di.univr.it/esame`

[2] <URL> server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo`