UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Algoritmi e strutture dati:

Homework 3

Anno Accademico 2023/2024

Salvatore Santoro M63001420
Andrea Russo M63001634

Problema 1

Traccia:

La S.S.C. Napoli è interessata a massimizzare i guadagni derivanti dagli ingressi allo stadio. Ti hanno assunto come consulente per aumentare le vendite dei "biglietti per gruppi" di tifosi. Bisogna risolvere il seguente problema: quando un gruppo vuole vedere una partita, tutti i membri del gruppo devono avere un posto, altrimenti se ne vanno. Quindi, poiché non vi possono essere gruppi "parziali", le gradinate spesso non sono piene. C'è ancora spazio disponibile, ma non abbastanza per un intero gruppo. In tal caso, il gruppo non può sedersi, e il Napoli perde soldi. Si sviluppi un algoritmo per decidere quali gruppi far entrare per massimizzare le vendite. Si alleghi al PDF un file editabile riportante l'implementazione in un linguaggio a scelta, corredato da almeno tre casi di test con il corrispondente output atteso.

Soluzione:

Dato un array in ingresso "arr" lungo n contenente il numero di persone che compongono ciascun gruppo il problema può essere risolto in tempo $O(nS)$ con un algoritmo di programmazione dinamica a suffissi con stato (knapsack con S = capacità massima dello stadio) associando ad ogni gruppo un valore pari ad 1(biglietto per gruppo) e un peso pari al numero di persone che lo compongono. In realtà, notando che ogni gruppo vale 1 indipendentemente dal numero di componenti (peso) è sufficiente far entrare sempre i gruppi più piccoli per massimizzare il numero di biglietti per gruppo venduti. Supponendo di ordinare l'array dei gruppi in ordine crescente di peso basta quindi effettuare una scelta greedy facendo entrare sempre il gruppo più piccolo disponibile fino a riempimento della capacità.

Complessità Temporale:

Abbiamo 3 alternative possibili con complessità temporali diverse:

- DP-Knapsack complessità $O(nS)$ con S dimensione dello stadio.
- Greedy con ordinamento effettuato con counting-sort(dato che i valori di persone per gruppo sono numeri interi) di complessità $O(N+k)$ dove k è il numero massimo di persone in un gruppo.
- Greedy con ordinamento $O(n \log n)$ che è la versione implementata, dato che Python utilizza un algoritmo di sorting (Timsort) che è $O(n \log n)$.

Da notare che le complessità considerate non tengono conto del tempo per scorrere l'array dopo averlo ordinato perché è al più $O(n)$. Ovviamente con $k = O(n)$ la soluzione con counting-sort risulta essere la più efficiente, $O(n)$.