Algoritmi e Strutture Dati 2

Anno Accademico: 2021-2022

Primo Homework

Docente: Francesco Pasquale

19 novembre 2021

Consegna: 26 novembre 2021 ore 19:00

Modalità di consegna. Ogni studente deve consegnare un unico file compresso (possibilmente in formato `zip`) contenente un file con l'elaborato (possibilmente in formato `pdf`) e i sorgenti dei programmi. Il file va inviato per posta elettronica a pasquale@mat.uniroma2.it entro le ore `19:00` di venerdì 26 novembre 2021. Inserire nella mail nome, cognome e numero di matricola.

Collaborazioni. È consentita e incoraggiata la collaborazione fra gli studenti al fine di risolvere gli esercizi. Tuttavia ogni studente deve poi scrivere il proprio elaborato individualmente e in modo autonomo.

Consigli. Scrivere le soluzioni in modo chiaro e conciso. Una soluzione corretta ma spiegata in modo poco chiaro o eccessivamente prolisso non prende il punteggio massimo. Viceversa, anche una soluzione non corretta può prendere qualche punto se presentata in modo ragionato.

Esercizio 1. Dato il problema Max Cut

INPUT: Un grafo $G = (V, E)$ con archi pesati $w : E \to \mathbb{N}$

SOLUZIONE: Una bipartizione dei nodi in due insiemi S e $\overline{S}$

OBIETTIVO: $\max \sum_{\{\{u,v\} \in E \,:\, u \in S,\, v \in \overline{S}\}} w(\{u, v\})$

Si consideri l'algoritmo seguente

Algorithm 1

Sia $V = \{v_1, v_2, \ldots, v_n\}$ l'insieme dei nodi
Inizializza $S_1 = \{v_1\}$ e $S_2 = \{v_2\}$
for $i = 3, \ldots, n$ **do**
  if $\sum_{\{u \in S_1 \,:\, \{u,v_i\} \in E\}} w(\{u, v_i\}) \leqslant \sum_{\{u \in S_2 \,:\, \{u,v_i\} \in E\}} w(\{u, v_i\})$ **then**
    Aggiungi v_i a S_1
  else
    Aggiungi v_i a S_2
return (S_1, S_2)

1. Dare un esempio di un'istanza su cui l'Algoritmo 1 non trova una soluzione ottima;
2. Mostrare che l'Algoritmo 1 è approssimante e stimare il fattore di approssimazione;
3. Mostrare che la vostra stima del fattore di approssimazione non può essere migliorata.

Esercizio 2. Dato un insieme finito di interi positivi $A \subset \mathbb{N}$, indichiamo con $s(A)$ la somma di tutti i numeri in A. Progettare un algoritmo che prenda in input un insieme finito di interi positivi $X \subset \mathbb{N}$ e restituisca in output due sottoinsiemi A e B di X, non vuoti e disgiunti, tali che $s(A) \geqslant s(B)$ e il rapporto $s(A)/s(B)$ sia minimo. Discutere correttezza ed efficienza dell'algoritmo.

Esercizio 3. Nel problema *Load Balancing* abbiamo $n \in \mathbb{N}$ *task* che devono essere eseguiti da $m \in \mathbb{N}$ macchine. Le macchine sono tutte equivalenti e il tempo impiegato da una macchina a eseguire il task i-esimo è $t_i \in \mathbb{N}$. Se a una macchina $h \in [m]$ viene assegnato il sottoinsieme $S \subseteq [n]$ dei task, allora la macchina h terminerà l'esecuzione dei task al tempo $L_h = \sum_{i \in S} t_i$. Vogliamo assegnare i task alle macchine in modo da minimizzare il massimo degli L_h.

1. Implementare il seguente algoritmo *greedy* in un linguaggio di programmazione a piacere:

   - Ordina i task in senso non crescente $t_1 \geqslant t_2 \geqslant \cdots \geqslant t_n$
   - Inizializza $L_h = 0$ per ogni macchina $h = 1, \ldots, m$
   - For $i = 1, \ldots, n$:

     Assegna il task i-esimo a una macchina h con L_h minimo e aggiorna $L_h = L_h + t_i$

   Il programma deve leggere l'input da un file di testo `input.txt` in cui nella prima riga compaiono due numeri interi n e m separati da uno spazio che rappresentano rispettivamente il numero di task e il numero di macchine; nelle n righe successive compaiono le durate dei task, una per ogni riga. Il programma deve scrivere un file di testo `output.txt` contenente il valore della soluzione trovata dall'algoritmo. Per esempio, se il file `input.txt` è

   $$\begin{array}{ll} 5 & 2 \\ 3 & \\ 3 & \\ 2 & \\ 2 & \\ 2 & \end{array} \tag{1}$$

   il file `output.txt` deve contenere 7.

2. Implementare un algoritmo esaustivo che trovi sempre una soluzione ottima e ne confronti il valore con quello della soluzione trovata dell'algoritmo greedy restituendo il fattore di approssimazione. Per esempio, se l'input è quello in (1), il valore della soluzione dell'algoritmo greedy è 7 mentre il valore di una soluzione ottima è 6, il programma quindi deve restituire 1.17.

3. Scrivere un programma che generi istanze casuali del problema *Load Balancing* e le usi per valutare empiricamente il rapporto di approssimazione medio dell'algoritmo greedy.[1]

[1] Scegliere a caso m, n e $t_1, \ldots t_n$; eseguire il programma con l'algoritmo greedy e il programma con l'algoritmo esaustivo sull'istanza ottenuta; ripetere un numero sufficientemente grande di volte l'esperimento e calcolare la media dei fattori di approssimazione ottenuti