Appunti *ordinati* sull'analisi degli algoritmi in preparazione all'orale

(Si può fare meglio di così?)

Emanuele Nardi

Compilato 25 gennaio 2019
v1.0.0

Indice

Elenco delle figure

Parte I
Laboratorio

0.1 Somma di due numeri (`somma`)

Dati due interi A e B, calcolare la loro somma.

```cpp
int main () {
    int N, M;
    ifstream in("input.txt");
    in >> N >> M;
    ofstream out("output.txt");
    out << N + M << "\n";

    return 0;
}
```

Commento `fstream` contiene le librerie di C++ per leggere e scrivere da file. L'input file stream (`ifstream`) si occupa di saltare tutti gli spazi ed i caratteri di nuova linea.

Formato dell'input La prima ed unica riga del file di input contiene i due interi A e B separati da spazio.

Formato dell'output L'output deve contenere un intero uguale alla somma di A e B.

0.2 Sottosequenza di somma massima (`sottoseq`)

Data una sequenza di interi $A[1 \dots N]$, vogliamo scegliere una sottosequenza $A[I \dots J]$ tale che la somma dei propri elementi sia massima. Come output chiediamo la somma di tale sottosequenza.

Formato dell'input La prima riga del file di input contiene l'intero N, il numero di elementi di A. Le successive N righe contengono un elemento di A, da A_1 a A_N .

Formato dell'output L'output deve contenere un intero uguale al valore della sottosequenza di somma massima.

```cpp
int N;
int array[1000000]; // dichiarato con la dimensione massima

int main() {
    ifstream in("input.txt");
    in >> N;
    for (int i = 0; i < N; i++)
        in >> array[i];

    int mx = 0; // soluzione parziale

    // per ogni sottosequenza
    for (int i = 0; i < N; i++) {
        for (int j = i; j < N; j++) {

            // calcola somma della sottosequenza
            int tot = 0;
```

```
            for (int k = i; k <= j; k++)
                tot += array[k];

            mx = max(mx, tot);
        }
    }

    ofstream out("output.txt");
    out << mx << endl;
    return 0;
}
```

```
int main () {
    ifstream in("input.txt");
    ofstream out("output.txt");

    int N;
    int mx = -1;
    int last = -1;
    int cur;

    in >> N;

    for (int i = 0; i < N; i++) {
        in >> cur;
        last = max(cur,cur+last);
        mx = max(mx,last);
    }

    out << mx << endl;

    return 0;
}
```

cur	last = max(cur,cur+last);	last	mx = max(mx,last);	mx
3	max(3, 2)	3	max(-1, 3)	3
-2	max(-2,1)	1	max(3, 1)	3
4	max(4,1)	4	max(3, 4)	4
1	max(1,5)	5	max(4, 5)	5
5	max(5,10)	10	max(5,1 0)	10

Tabella 1: Tabella di calcolo sul secondo esempio della descrizione del problema

0.3 Sottomatrice di somma massima (`sottomat`)

Data una matrice di interi da R righe e C colonne, trovare il rettangolo al suo interno di somma massima. Stamparne la somma.

Formato dell'input La prima riga del file di input contiene due interi, R e C, rispettivamente il numero di **righe** e di **colonne** della matrice. Le successive R righe contengono ognuna C interi separati da spazio. L'i-esimo intero sulla j-esima riga corrisponde al valore della matrice sulla riga i e colonna j.

Formato dell'output L'output deve contenere un intero uguale alla somma della sottomatrice di somma massima.

```
int R, C;

int A[1001][1001]; // matrice con i valori
int S[1001][1001]; // matrice di appoggio
// S[i][j] é uguale alla somma della riga i fino all'elemento j

int main(void) {
    ifstream in("input.txt");
    ofstream out("output.txt");

    in >> R >> C;

    for (int i = 1; i <= R; i++) {
        // mantiene la somma della riga corrente
        int sumr = 0;
        for (int j = 1; j <= C; j++) {
        // scorro le colonne

            in >> A[i][j];   // prende in input il numero
            sumr += A[i][j]; // lo somma alla somma parziale della riga corrente
            S[i][j] = sumr;  // memorizza la somma paziale nella matrice di aiuto
        }
    }

    int sol = -1;
    // per ogni coppia di colonne
    for (int c1 = 1; c1 <= C; c1++) {
        for (int c2 = c1; c2 <= C; c2++) {
            // visito la colonna con l'algoritmo lineare per la sottosequenza

            int tot = 0;
            for (int r = 1; r <= R; r++) {
                int cur = S[r][c2] - S[r][c1-1];
                tot = max(cur,cur+tot);
                sol = max(sol,tot);
            }
        }
    }

    out << sol << endl;

    return 0;
}
```

$$\begin{pmatrix} 2 & -9 & 2 & 3 \\ 1 & 4 & 5 & 1 \\ -2 & 3 & 4 & 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 2 & -7 & -5 & -2 \\ 1 & 5 & 10 & 11 \\ -2 & 1 & 5 & 6 \end{pmatrix}$$

Figura 1: Rappresentazione della matrice di partenza (a sinistra) e di quella che contiene le somme parziali (a destra)

0.4 Ordinamento (`mergeSort`)

Dato un array di interi, ordinatelo. Siete liberi di utilizzare le librerie, ma implementare un algoritmo di ordinamento visto a lezione può essere un esercizio più utile.

Formato dell'input La prima riga del file di input contiene N, il *numero di interi nell'array*. La seconda riga contiene l'array, con gli elementi separati da spazi.

Formato dell'output L'output deve contenere gli elementi ordinati dell'array, separati da uno spazio.

0.5 Intervalli (`intervalli`)

Vi vengono dati una serie di N intervalli temporali, ognuno rappresentati come una coppia $Inizio_i$, $Fine_i$ di interi. Vogliamo sapere quale sia il più lungo periodo non coperto da alcun intervallo, considerando solo gli istanti compresi fra il minimo istante di inizio ed il massimo istante di fine degli intervalli.

Formato dell'input La prima riga del file di input contiene l'intero N, il numero di intervalli. Le successive N righe contengono due interi separati da spazio, l'istante di inizio e quello di fine dell'intervallo.

Formato dell'output L'output deve contenere due interi che rappresentano l'istante di inizio e quello di fine del più lungo periodo scoperto. Se ci sono più di un periodo scoperto della stessa lunghezza, restituire quello con inizio minore.
Il Diagramma di Gantt formato dagli intervalli di esempio è mostrato in figura 2.

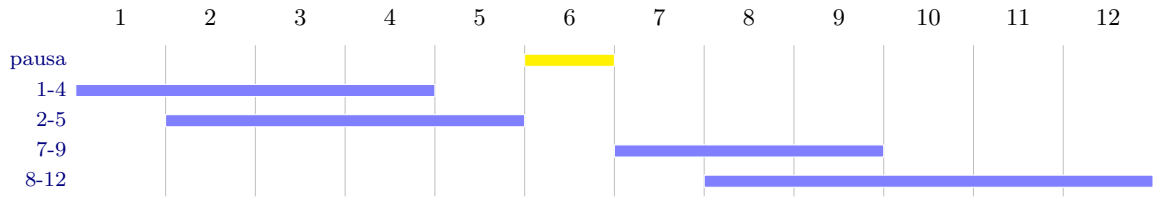


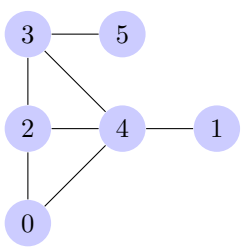
Figura 2: Gantt Chart sorted

0.6 Ordinamento pesato (`sortpesato`)

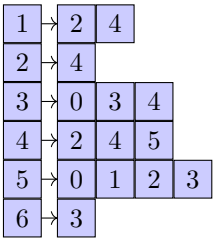
Vi viene dato un array di N interi da ordinare. Gli elementi sono tutti diversi, anzi sono precisamente tutti gli interi fra 1 e N. Visto che sarebbe troppo facile ordinare un array del genere, abbiamo delle restrizioni. Ad ogni turno potete scambiare due elementi a scelta dell'array. Per fare ciò, pagate un prezzo pari alla somma dei due elementi. Per scambiare di posto l'elemento 3 e l'elemento 4 impiegate un turno e pagate 7. Voi dovete risolvere due problemi: qual è il metodo più veloce (che ottimizza il numero di turni) ed il metodo più economico (che ottimizza il prezzo).

Commento Il metodo più veloce è un quickSort o un mergeSort (ossia un algoritmo di ordinamento ottimale). Invece il metodo meno costoso potrebbe essere quello di muovere l'intero più piccolo.

Formato dell'input La prima riga del file di input contiene N, la lunghezza dell'array. La riga successiva contiene l'array, con gli elementi separati da uno spazio.



(a) Rappresentazione grafica



(b) Rappresentazione tramite lista di adicenza

Figura 3: Grafo orientato

Formato dell'output L'output deve contenere due interi. Il primo intero S rappresenta il numero minimo di turni per ottenere l'array ordinato. Il secondo intero P rappresenta il prezzo minimo per ordinare l'array.

0.7 Visita di un grafo orientato (`visita`)

Dato un grafo orientato e un nodo di partenza S, calcolate il numero di nodi raggiungibile da S.

Formato dell'input La prima riga del file di input contiene tre interi, N, M e S. N è il numero di nodi, M il numero di archi, S il nodo di partenza. Le successive M righe contengono **due interi per riga**, il nodo di partenza e di arrivo degli archi.

Formato dell'output L'output deve contenere un intero, uguale al numero di nodi raggiungibili da S (S incluso).

0.8 Diametro di un grafo non orientato (`diametro`)

Vi viene dato in input un grafo non orientato. Dovete trovare la coppia di nodi più lontani e stamparne la distanza.

Formato dell'input La prima riga del file di input contiene due interi, N e M. N è il numero di nodi, M il numero di archi. Le successive M righe contengono due interi per riga, i due nodi collegati dall'arco.

Formato dell'output L'output deve contenere un intero, uguale alla massima distanza fra due nodi.

Riferimenti utili

- `front()`

0.9 Numero di cammini minimi (`numcammini`)

Vi viene dato in input un grafo orientato ed una coppia di nodi S e T. Dovete trovare quanti diversi cammini di lunghezza minima ci sono fra S e T.

Formato dell'input La prima riga del file di input contiene quattro interi, N, M, S e T. N è il numero di nodi, M il numero di archi, S il nodo di partenza e T il nodo di arrivo. Le successive M righe contengono due interi per riga, il nodo di partenza e di arrivo degli archi.

Formato dell'output Il file di output deve contenere due interi separati da spazio. Il primo è uguale alla distanza fra S e T, il secondo al numero di percorsi di lunghezza minima da S a T.

0.10 Luddisti Spaziali (`space`)

Testo del problema

0.11 Componente fortemente connessa (`componente`)

Dato un grafo orientato, trovare la dimensione della componente fortemente connessa di dimensione massima. Un insieme di nodi forma una componente fortemente connessa se esiste un percorso fra ogni coppia di nodi in entrambe le direzioni.

Formato dell'input La prima riga del file di input contiene due interi, N e M. N è il numero di nodi, M il numero di archi. Le successive M righe contengono ognuna due interi. L'i-esima riga contiene la sorgente e la destinazione dell'i-esimo arco.

Formato dell'output Il file di output deve contenere un intero pari alla dimensione della più grande componente fortemente connessa

0.12 Ordinamento topologico (`toporder`)

Dato un grafo orientato aciclico, stampare un suo ordinamento topologico.

Formato dell'input La prima riga del file di input contiene due interi, N e M. N è il numero di nodi, M il numero di archi. Le successive M righe contengono ognuna due interi. L'i-esima riga contiene la sorgente e la destinazione dell'i-esimo arco.

Formato dell'output La *prima ed unica* riga del file di output deve contenere N interi separati da spazio, l'ordinamento topologico trovato dall'algoritmo.

0.13 Cammino lungo (`camminolungo`)

Dato un grafo orientato aciclico, trovare il cammino più lungo nel grafo.

Formato dell'input La prima riga del del file di input contiene due interi, N e M. N è il numero di nodi, M il numero di archi. Le successive M righe contengono ognuna due interi. L'i-esima riga contiene la sorgente e la destinazione dell'i-esimo arco.

Formato dell'output Il file di output consiste di un unica riga contenente un intero pari al cammino di lunghezza massima

0.14 Batman si annoia (`batman`)

Testo del problema .

0.15 Zaino (`zaino`)

Avete uno zaino di capacità C ed un insieme di N oggetti. Per ognuno di questi oggetti, sapete il peso (P_i) e il valore (V_i). Dovete scegliere quali oggetti mettere nello zaino in modo da ottenere il maggior valore possibile senza superare la capacità dello zaino.

Formato dell'input La prima riga del file di input contiene due interi, C e N. Le successive N righe contengono due interi ognuna, P_i e V_i.

Formato dell'output Il file di output deve contenere un intero: il massimo valore che è possibile trasportare con lo zaino.

0.16 Sottoinsieme crescente di somma massima (`sottocres`)

Vi viene dato in input un array di N interi $A_1 \dots A_N$. Per ogni elemento potete scegliere se includerlo nell'insieme soluzione, o se ignorarlo. Se un elemento A_i fa parte dell'insieme, tutti gli elementi che lo succedono nell'array e che hanno valore minore di A_i non possono essere inclusi nell'insieme. In altre parole, gli elementi dell'insieme, quando stampati nell'ordine in cui si trovavano nell'array, devono formare una sequenza crescente. Vogliamo massimizzare la somma degli elementi dell'insieme.

Formato dell'input La prima riga del file di input contiene un intero N. La seconda riga contiene N interi separati da spazio.

Formato dell'output Il file di output deve contenere un intero, la somma degli elementi dell'insieme di somma massima.

0.17 Pillole (`pillole`)

La zia Lucilla deve assumere ogni giorno mezza pillola di una certa medicina. Lei inizia il trattamento con una bottiglia che contiene esattamente N pillole.
Durante il primo giorno lei prende una pillola dalla bottiglia, la spezza in due, ne ingerisce una metà e rimette l'altra metà nella bottiglia.
Nei giorni seguenti lei prende un pezzo a caso della bottiglia (potrebbe essere una pillola intera o una mezza pillola). Se ha pescato una mezza pillola la ingerisce. Se ha pescato una pillola intera la spezza a metà, rimette una delle due mezze pillole nella bottiglia e ingerisce l'altra mezza pillola.
La zia può svuotare la bottiglia in tanti modi diversi. Rappresentiamo la cura come una stringa di $2N$ caratteri, in cui il carattere i-esimo è "I" se la zia ha pescato una pillola intera nel giorno i e "M" se la zia ha invece pescato una mezza pillola. Nel caso in cui la bottiglia originaria contenga 3 pillole intere, le possibili sequenze sono le seguenti:

1. `IIIMMM`
2. `IIMIMM`
3. `IIMMIM`
4. `IMIIMM`
5. `IMIMIM`

Il problema vi richiede di scrivere un programma che, dato N, restituisca il numero di possibili sequenze nel trattamento.

Formato dell'input Il file di input consiste di un unica linea contenente l'intero N, il numero di pillole presenti nella bottiglia all'inizio della cura.

Formato dell'output Il file di output contiene un unico intero, il numero di diversi modi in cui la zia finisce la bottiglia.

Assunzioni L'output sarà abbastanza piccolo da poter essere mantenuto dentro un `long long int`.

0.18 I travestimenti di Sherlock Holmes (`scherlock`)

Testo del problema .

0.19 Sottosequenza comune massimale (`lcs`)

Definiamo come sottosequenza di una stringa una qualunque sequenza di caratteri ottenibile partendo dalla stringa di partenza, ed eliminando 0 o più caratteri e senza cambiarne l'ordine. Il nostro obiettivo è quello di trovare, date due stringhe, la più lunga sottosequenza in comune e stamparne la lunghezza.

Formato dell'input L'input consiste di due righe, ogni riga contiene una delle due stringhe.

Formato dell'output Il file di output contiene un intero, la lunghezza della sottosequenza comune massimale.

0.20 Min-Cover su albero (`mincover`)

Dato un albero, un sottoinsieme dei suoi nodi S è un Node-Cover se, per ogni arco dell'albero, uno dei suoi due nodi fa parte di S. Dato un albero, trovare il Node-Cover con il minimo numero di nodi e stamparne la dimensione.

Formato dell'input La prima riga del file di input contiene N, il numero di nodi. Le successive $N-1$ contengono ognuna una coppia P_i, F_i, una coppia padre-figlio.

Formato dell'output Il file di output deve contenere un intero, la dimensione del Node-Cover

0.21 Min-Cover su albero pesato (`mincoverpesato`)

Dato un albero, un sottoinsieme dei suoi nodi S è un Node-Cover se, per ogni arco dell'albero, uno dei suoi due nodi fa parte di S. Vi viene dato in input un albero con dei pesi sui nodi. Trovare il Node-Cover con il peso minimo e stamparne la dimensione.

Formato dell'input La prima riga del file di input contiene N, il numero di nodi. La riga successiva contiene N interi separati da spazio. L'i-esimo intero rappresenta il peso del nodo i. Le successive $N-1$ contengono ognuna una coppia P_i, F_i, una coppia padre-figlio.

Formato dell'output Il file di output deve contenere un intero, il peso del Node-Cover di peso minimo.

0.22 Fiera (`fiera`)

Testo del problema