

Luiss

Libera Università Internazionale
degli Studi Sociali Guido Carli

Corso di preparazione per la selezione territoriale delle Olimpiadi di Informatica

Lezione 4. Algoritmi greedy

Irene Finocchi

15 aprile 2021

LUISS 





Preferibile se hai problemi di connessione, ma non permette di condividere le tue soluzioni

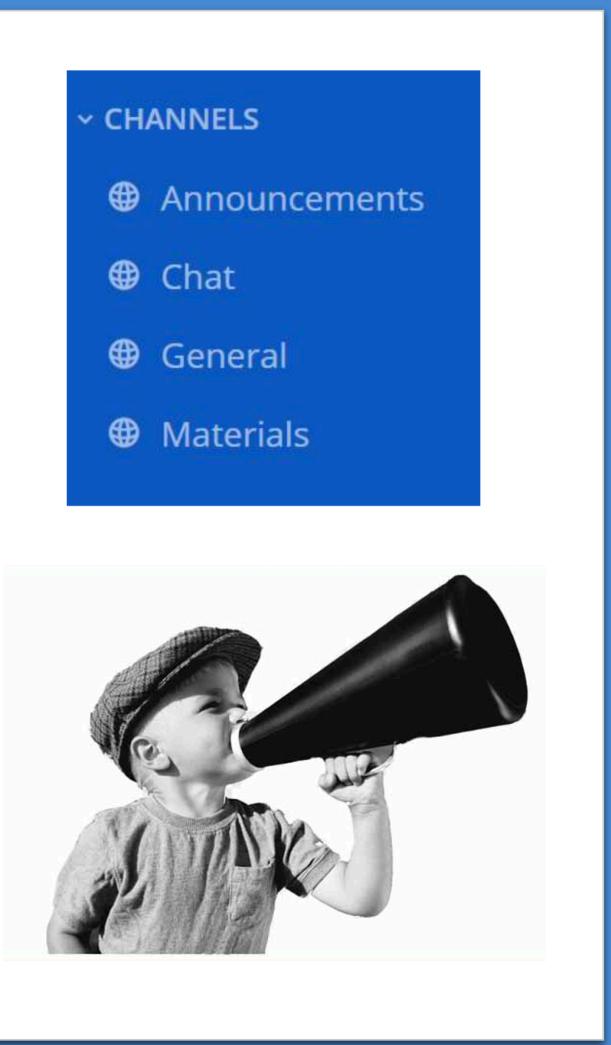
Lezioni

Puoi accedere alle lezioni tramite YouTube o Webex



Licenza limitata a 1000 utenti; puoi provare l'ebrezza di diventare **panelist** e condividere la tua soluzione ai problemi considerati!





The image shows a screenshot of a Mattermost application. On the left, there's a sidebar with a blue header labeled "CHANNELS". Underneath, four channels are listed with icons: "Announcements" (globe), "Chat" (globe), "General" (globe), and "Materials" (globe). Below this sidebar is a white rectangular area containing a photograph of a young boy wearing a cap and a grey t-shirt. He is holding a large black megaphone to his mouth, looking upwards and to the side as if shouting.

Repetita iuvant (sed stufant!)

- Non avrai altra chat al di fuori di **mattermost**
- <https://coding.luiss.ml/mattermost/> (canale Chat)
 - Usa il canale Chat per le domande
 - Potete rispondere anche voi!
- Controlla i canali: spesso troverai qui le risposte alle tue domande!
- Non inquinare i canali! Contribuisci a mantenerli puliti e non postare contenuti impropri. Grazie!
- Le lezioni si svolgono sempre nell'orario 16:00-19:00

Programma di oggi

1. Soluzione esercizi assegnati la scorsa settimana
 - Due **soluzioni guidate**: **Domino massimale** e **Treno di container**
 - Le vostre soluzioni per gli altri tre problemi
2. La tecnica greedy
 1. **Problemi di sequenziamento**
 2. **Problemi su intervalli**: **Irrigazioni** (Luiss contest 2020), **Giri sulla scopa** **Nimbus3000** (Territoriali 2007), **Turni di guardia** (Territoriali 2012)
 3. **Huffman e applicazioni**: bonifici (Luiss contest 2020)
3. (Un problema dalle Territoriali 2010: Sbarramento tattico)
4. Esercizi per casa

Due problemi risolti

- Domino massimale (Territoriali 2011)
<https://training.olinfo.it/#/task/domino/statement>
- Treno di container (Territoriali 2009)
<https://training.olinfo.it/#/task/treno/statement>

Un problema dalle Territoriali 2011: Domino massimale

<https://training.olinfo.it/#/task/domino/statement>



Selezioni territoriali 2011

Domino massimale (domino)

Difficoltà D = 2.



Descrizione del problema

Sono date N tessere di domino, dove ogni tessera contiene due numeri compresi tra 0 e 6 in corrispondenza delle sue due estremità. Le tessere possono essere ruotate e la regola impone che due tessere possono essere concatenate se le loro estremità in contatto hanno inciso lo stesso numero. Aiuta a trovare il maggior numero di tessere che si possono concatenare a formare un'unica catena: non è detto che si riescano sempre a usare tutte le tessere; inoltre, possono esserci due o più tessere uguali a meno di rotazioni.

Dati di input

Il file `input.txt` è composto da $N + 1$ righe. La prima riga contiene l'intero positivo N , il numero delle tessere a disposizione. Ciascuna delle successive N righe contiene due interi positivi (compresi tra 0 e 6) separati da una spazio, che rappresentano i numeri incisi sulle estremità delle tessere.

Dati di output

Il file `output.txt` è composto da una sola riga contenente il massimo numero di tessere che possono essere concatenate con le regole del domino.

Assunzioni

- $2 \leq N \leq 10$.

Esempi di input/output

File input.txt

```
6  
3 0  
4 0  
2 6  
4 4  
0 1  
1 0
```

File output.txt

```
5
```

Nota/e

- In generale, più configurazioni possono soddisfare i requisti del problema: è sufficiente fornire la lunghezza massima.

Permutazioni ricorsive: intuizione

$x_1 | x_2 | x_3 | x_4$

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_2	x_4	x_1	x_3

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_3	x_4	x_2	x_1

$x_1 | x_2 | x_3 | x_4$

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

$x_1 | x_2 | x_3 | x_4$

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

$x_1 | x_2 | x_3 | x_4$

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generare le permutazioni ricorsivamente

```
1 int N;
2 struct tessera{
3     int s,d;
4 };
5 tessera t[10];
6 tessera r[10];
7 bool usata[10];
8 tessera permutazione[10];
9 int lunghezza = 0;
```

Generare le permutazioni ricorsivamente

```
34 int main()
35 {
36     ifstream in("input.txt");
37     ofstream out("output.txt");
38     in >> N;
39     for(int i = 0; i < N; i++)
40     {
41         in >> t[i].s >> t[i].d;
42         r[i].s = t[i].d;
43         r[i].d = t[i].s;
44     }
45     for(int i = 0; i < N; i++)
46         usata[i] = false;
47     trova_permutazione(0);
48     out << lunghezza;
49     return 0;
50 }
```

Generare le permutazioni ricorsivamente

```
11 void trova_permutazione(int pos)
12 {
13     if (pos > lunghezza)
14         lunghezza = pos;
15     for (int i = 0; i < N; i++)
16     {
17         if (pos == 0 || (permutazione[pos-1].d == t[i].s && usata[i] == false))
18         {
19             permutazione[pos] = t[i];
20             usata[i] = true;
21             trova_permutazione(pos + 1);
22             usata[i] = false;
23         }
24         if (pos == 0 || (permutazione[pos-1].d == r[i].s && usata[i] == false))
25         {
26             permutazione[pos] = r[i];
27             usata[i] = true;
28             trova_permutazione(pos + 1);
29             usata[i] = false;
30         }
31     }
32 }
```

Una soluzione più idiomatica in C++ (non ricorsiva)

```
const int MAXN = 10;
int N;
vector< pair< int, int > > tessere;

int conta(int prec) {
    int i = 1;
    while (true) {
        if (i == N) break;
        else if (prec == tessere[i].first) {
            prec = tessere[i].second;
            ++i;
        }
        else if (prec == tessere[i].second) {
            prec = tessere[i].first;
            ++i;
        }
        else break;
    }
    return i;
}
```

Una soluzione più idiomatica in C++ (non ricorsiva)

```
int main(void) {
    in >> N;
    for (int i = 0; i < N; i++) {
        int a, b;
        in >> a >> b;
        tessere.push_back(make_pair(a, b));
    }

    int sol = 1;

    sort(tessere.begin(), tessere.end());
    while (next_permutation(tessere.begin(), tessere.end())) {
        sol = max(sol, conta(tessere[0].first));
        sol = max(sol, conta(tessere[0].second));
    }

    out << sol << endl;
    return 0;
}
```

Un problema dalle Territoriali 2009: Treno di container

<https://training.olinfo.it/#/task/treno/statement>



Treno di container (treno)

Difficoltà D = 2.

Descrizione del problema

Al porto sono arrivati N container della sostanza chimica di tipo A e N container della sostanza chimica di tipo B. I container sono stati caricati, uno dietro l'altro, su di un treno che ne può contenere $2N+2$. Le posizioni dei container sul treno sono numerate da 1 a $2N+2$. Il carico è stato fatto in modo che gli N container di tipo A occupino le posizioni da 1 a N , mentre quelli di tipo B da $N+1$ a $2N$; le rimanenti due posizioni $2N+1$ e $2N+2$ sono vuote.

Per motivi connessi all'utilizzo delle sostanze chimiche nella fabbrica alla quale sono destinate, i container vanno distribuiti sul treno a coppie: ciascun container per la sostanza di tipo A deve essere seguito da uno di tipo B. Occorre quindi che nelle posizioni dispari ($1, 3, 5, \dots, 2N-1$) vadano sistemati esclusivamente i container di tipo A mentre in quelle pari ($2, 4, 6, \dots, 2N$) quelli di tipo B, lasciando libere le ultime due posizioni $2N+1$ e $2N+2$.

A tal fine, viene impiegata una grossa gru, che preleva due container alla volta, in posizioni consecutive $i, i+1$, e li sposta nelle uniche due posizioni consecutive $j, j+1$ libere nel treno (inizialmente, $j = 2N+1$). Tale operazione è univocamente identificata dalla coppia (i, j) , dove entrambe le posizioni i e $i+1$ devono essere occupate da container mentre j e $j+1$ devono essere entrambe vuote.

Per esempio, con $N = 4$, abbiamo inizialmente la configurazione A A A A B B B B * *, dove le due posizioni vuote sono indicate da un asterisco *:

- Il primo spostamento della gru è (4,9) e porta alla configurazione:

A A A * * B B B A B
1 2 3 4 5 6 7 8 9 10

- Il secondo spostamento è (6, 4) e porta alla configurazione:

A A A B B * * B A B
1 2 3 4 5 6 7 8 9 10

- Il terzo spostamento è (2, 6) e porta alla configurazione:

A * * B B A A B A B
1 2 3 4 5 6 7 8 9 10

- Il quarto spostamento è (5,2) e porta alla configurazione:

A B A B * * A B A B
1 2 3 4 5 6 7 8 9 10

- Il quinto e ultimo spostamento è (9,5) e porta alla configurazione desiderata:

A B A B A B A B * *
1 2 3 4 5 6 7 8 9 10

Notare che per $N=4$ è possibile, con cinque spostamenti, sistemare i $2N$ container nell'ordine giusto. Scrivere quindi un programma che determini la successione degli spostamenti eseguiti dalla gru per ottenere un analogo risultato nel caso in cui $3 \leq N \leq 1000$. Si richiede inoltre che il numero K di tali spostamenti non superi il valore $3N$.

Dati di input

Il file `input.txt` è composto da una sola riga, contenente l'intero N che rappresenta il numero di container per ciascuna delle due sostanze.

Dati di output

Il file `output.txt` è composto da $K+1$ righe.

La prima riga contiene due interi positivi separati da uno spazio, rispettivamente il numero K di spostamenti operati dalla gru e il numero N di container per ciascuna delle due sostanze

Le righe successive contengono la sequenza di K spostamenti del tipo (i,j) , tali che partendo dalla sequenza `AAA...ABBB...B**`, si arrivi alla sequenza `ABABAB...AB**` con le regole descritte sopra. Ciascuna delle righe contiene una coppia di interi positivi i e j separati da uno spazio a rappresentare lo spostamento (i,j) .

Assunzioni

- $3 \leq N \leq 1000$,
- $1 \leq i, j \leq 2N+1$,
- $K \leq 3N$.

Esempi di input/output

File input.txt	File output.txt
3	4 3 2 7 6 2 4 6 7 4
4	5 4 4 9 6 4 2 6 5 2 9 5

File input.txt

5

File output.txt

6 5
5 11
2 5
9 2
6 9
3 6
11 3

Da N a N-1... attenzione al caso base

```
int K, N;
ifstream in("input.txt");
ofstream out("output.txt");
void calcola(int j) {
    if (j==10) {
        out << (j-2)/2 << " " << j-1 << endl;
        return;
    }
    out << (j-2)/2 << " " << j-1 << endl;
    out << j-3 << " " << (j-2)/2 << endl;
    calcola(j-2);
}
```

Da N a N-1... attenzione al caso base

```
int main()
{
    K = 2*N-3;
    out << K << " " << N << endl;
    calcola(2*N+2);
    out << 6 << " " << 4 << endl;
    out << 2 << " " << 6 << endl;
    out << 5 << " " << 2 << endl;
    out << 2*N+1 << " " << 5 << endl;
    return 0;
}
```

Soluzioni dei rimanenti esercizi assegnati la scorsa settimana



Le vostre soluzioni

- Quasi-palindromi (Territoriali 2010)
<https://training.olinfo.it/#/task/quasipal/statement>
- Cerca le somme (GATOR 2015)
<https://training.olinfo.it/#/task/cercalesomme/statement>
- Fractal painting (OIS 2018)
https://training.olinfo.it/#/task/ois_painting/statement

La tecnica greedy



GREEDY
GREEDY



Problemi di ottimizzazione

- La tecnica greedy è usata per risolvere **problemi di ottimizzazione**
- Vogliamo trovare la **migliore soluzione a un problema**
 - La **strada più breve** per andare da Napoli a Torino
 - Il **prezzo più basso** per acquistare online 100 biglietti per un concerto
- Greedy (**avidio, goloso** in italiano) è una tecnica algoritmica molto efficiente per quei problemi in cui **una scelta localmente ottima porta a una soluzione ottima del problema**
 - Non sempre è così: vedremo un esempio!

Esempio: acquisto di biglietti

Si supponga di dover comprare *online* N biglietti per un concerto al minor prezzo possibile

Strategia:

1. Scelgo il sito dove i biglietti costano meno e compro tutti i biglietti possibili su quel sito
2. Se sono arrivato a N ho terminato, altrimenti scelgo il sito successivo in cui i biglietti siano meno cari

Osservazioni

1. In questo esempio dovrebbe essere evidente che la **soluzione** è quella **ottima**: compriamo prima i biglietti meno cari, fino ad esaurimento
2. Si fa una **scelta locale**: il sito che al momento ha il prezzo più basso
3. Implicitamente, stiamo usando un **ordinamento**: si parte dal sito con il prezzo minore e si va avanti (acquistando in siti via via più costosi) fino a comprare N biglietti

L'ordinamento di N oggetti si può fare in modo molto efficiente (mergesort, quicksort... usate il metodo sort della STL)

Gli elementi della tecnica greedy

1. Insieme di **candidati** (città o siti che vendono biglietti)
2. Insieme dei candidati già esaminati
3. Funzione **ammissibile**: verifica se un insieme di candidati rappresenta una soluzione (anche parziale), non necessariamente ottima (un insieme di città è un cammino da Napoli a Torino?)
4. Funzione **seleziona**: indica quale dei candidati non ancora esaminati è al momento il più promettente
5. **Funzione obiettivo** da minimizzare o massimizzare (lunghezza del cammino da Napoli a Torino o cifra spesa per acquistare N biglietti)

Idea di base: pseudocodice

algoritmo `paradigmaGreedy`(insieme di candidati C) \rightarrow soluzione

$S \leftarrow \emptyset$

while ((incompleta (S)) **and** ($C \neq \emptyset$)) **do**

$x \leftarrow \text{seleziona}(C)$

$C \leftarrow C - \{x\}$

if (ammissibile($S \cup \{x\}$)) **then** $S \leftarrow S \cup \{x\}$

return S

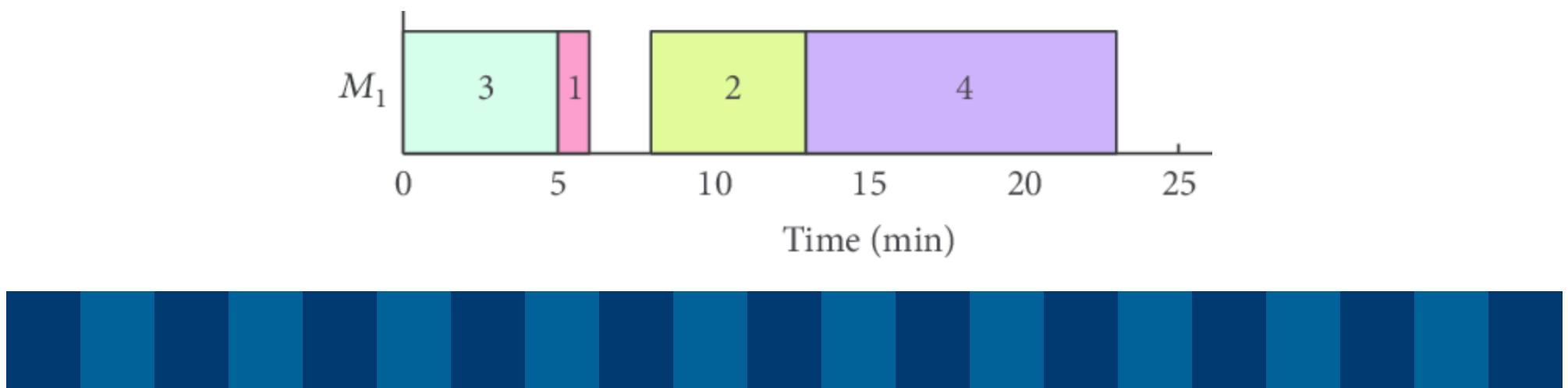
Perché goloso?

Perché sceglie sempre il candidato più promettente!



Esempio: greedy OK

Un problema di sequenziamento di lavori



Un problema di sequenziamento

- Un server (per esempio una CPU o un impiegato dell'ufficio postale) deve servire **n clienti**
- Il server può decidere in quale ordine servirli
- Il servizio richiesto dall'i-esimo cliente richiede t_i secondi
- Chiamiamo $T(i)$ il **tempo di attesa del cliente i**
 - $T(i) = \text{somma dei tempi richiesti per servire tutti i clienti precedenti e il cliente } i \text{ stesso}$
- Vogliamo minimizzare il **tempo di attesa medio**, ovvero:

$$\frac{T(1)+T(2)+T(3)+\dots+T(n)}{n}$$

Esempio

$$t_1 = 50 \text{ msec}, \quad t_2 = 100 \text{ msec}, \quad t_3 = 3 \text{ msec}$$

Dei 6 possibili ordinamenti, il migliore è evidenziato in **rosso**

<i>Ordine</i>	<i>T</i>			
1 2 3	$50 + (50 + 100) + (50 + 100 + 3)$	msec	=	353 msec
1 3 2	$50 + (50 + 3) + (50 + 3 + 100)$	msec	=	256 msec
2 1 3	$100 + (100 + 50) + (100 + 50 + 3)$	msec	=	403 msec
2 3 1	$100 + (100 + 3) + (100 + 3 + 50)$	msec	=	356 msec
3 1 2	$3 + (3 + 50) + (3 + 50 + 100)$	msec	=	209 msec
3 2 1	$3 + (3 + 100) + (3 + 100 + 50)$	msec	=	259 msec

Algoritmo greedy

- Il seguente algoritmo genera l'ordine di servizio in maniera incrementale secondo una strategia greedy
- Supponiamo di aver deciso di servire i clienti i_1, i_2, \dots, i_m . Se decidiamo di servire il cliente j , il tempo totale di servizio diventa:

$$t_{i_1} + t_{i_2} + \dots + t_{i_m} + t_j$$

- Quindi al generico passo l'algoritmo **serve la richiesta più corta tra quelle rimanenti**
- Tempo di esecuzione: **$O(n \log n)$** , per ordinare le richieste per lunghezze crescenti



Esempio: greedy KO

Un problema di cambio di denaro



Il problema e l'algoritmo greedy "naturale"

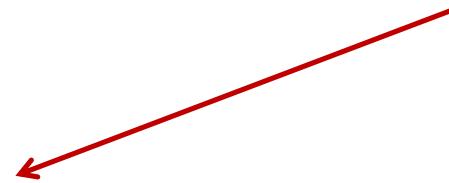
- Input: un numero intero positivo n
- Output: il più piccolo numero intero di banconote per cambiare n Euro usando pezzi da 20, 10, 5 e 1 Euro
- Esempi
 - $n = 58$, 7 banconote: $20+20+10+5+1+1+1$
 - $n = 18$, 5 banconote: $10+5+1+1+1$
- Algoritmo greedy
 - Dispensa una banconota alla volta
 - Ad ogni passo, utilizza la **banconota più grande che non superi la cifra rimanente**

Criterio di scelta
greedy

Un controesempio all'ottimalità

- Input: un numero intero positivo n
- Output: il più piccolo numero intero di banconote per cambiare n Euro usando pezzi da 12, 8 e 1 Euro
- Esempi
 - $n = 31$, greedy usa 9 banconote:
 $12+12+1+1+1+1+1+1+1$
 - $n = 31$, l'ottimo usa 6 banconote: $12+8+8+1+1+1$

Greedy non
garantisce
l'ottimalità



Problemi su intervalli

Irrigazioni (Luiss contest 2020)

Slide a cura di Carlo Malagnino!



Descrizione del problema

Il nocciolo di Irene contiene una lunga fila di N piante che richiedono di essere irrigate. L' i -esima pianta si trova a d_i metri di distanza dall'origine della fila, in cui è posizionato il rubinetto del pozzo.

Irene deve acquistare il materiale per realizzare l'impianto di irrigazione. Il negozio in cui si è recata dispone di tubi forati di lunghezza unitaria (1 metro) da inserire opportunamente lungo la canalizzazione, in corrispondenza delle piante, per consentirne l'irrigazione.

Aiuta Irene a capire qual è il minimo numero di tubi da acquistare per poter irrigare tutte le N piante minimizzando la spesa per i tubi (non è necessario calcolare le posizioni in cui vanno installati i tubi, ma solo il loro numero!).

Dati di input

La prima riga del file di input contiene un intero T , il numero di casi di test. Seguono T casi di test, numerati da 1 a T . Ogni caso di test è preceduto da una riga vuota.

Ciascun caso di test è composto da $N + 1$ righe: la prima contiene il numero intero N , e le N righe successive contengono le distanze d_i di ogni pianta dall'origine della fila.

Dati di output

Il file di output deve contenere la risposta ai casi di test che sei riuscito a risolvere. Per ogni caso di test che hai risolto, il file di output deve contenere una riga con la dicitura

```
Case #t: x
```

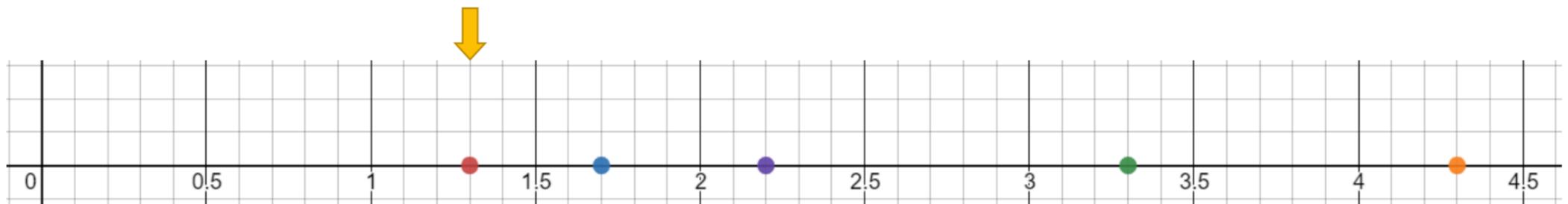
dove `t` è il numero del caso di test (a partire da 1) e `x` è il numero di tubi da acquistare.

Assunzioni

- $N \geq 2$.
- $N \leq 10^6$.
- $T \leq 25$.
- $d_i \leq 10^9$.
- I valori d_i sono numeri in virgola mobile.

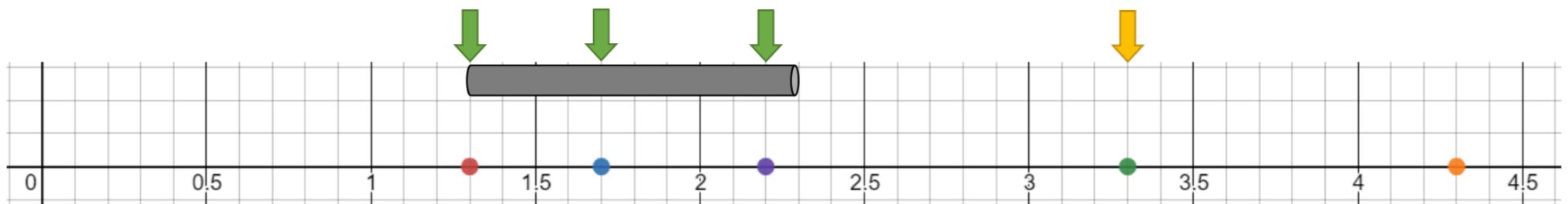
Esempio di Input/Output

Input
1
5
4.3
1.3
2.2
1.7
3.3



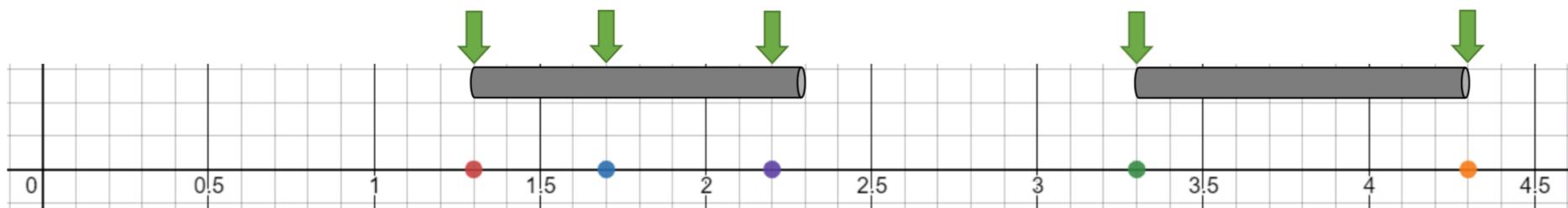
Esempio di Input/Output

Input
1
5
4.3
1.3
2.2
1.7
3.3



Esempio di Input/Output

Input
1
5
4.3
1.3
2.2
1.7
3.3



Output
Case #1: 2

Soluzione in C++

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    int T;
    cin >> T;

    for(int t=1;t<=T;t++) {
        long long int N;
        cin >> N;
        vector<double>A(N);
        for(long long int i=0;i<N;i++) {
            cin >> A[i];
        }
        sort(A.begin(),A.end());
        double mas=0;
        long long int ris=0;
        for(long long int i=0;i<N;i++) {
            if(mas<A[i]) {
                ris++;
                mas=A[i]+1;
            }
        }
        cout << "Case #" << t << ":" << ris << "\n";
    }
}
```

Turni di guardia (Territoriali 2012)

<https://training.olinfo.it/#/task/turni/statement>



Turni di guardia (turni)

Difficoltà D = 2.

Descrizione del problema

La Banda Bassotti è stata rimessa in libertà. Zio Paperone, in partenza per un viaggio di K giorni, ha la necessità di far sorvegliare il deposito: quindi ha bisogno che sia sempre presente almeno una persona. Per risparmiare, decide di chiedere la disponibilità di amici e parenti, e ognuno di questi fornisce un intervallo di giorni in cui è disponibile per la sorveglianza. Paperone però sa che dovrà fare un regalo a ognuna delle persone che userà, e volendo risparmiare al massimo deve coinvolgere il minimo numero di persone, senza lasciare mai il deposito scoperto. In questo modo riuscirà a risparmiare sui regali.

Per esempio, supponiamo che il viaggio di Zio Paperone sia di K=8 giorni, con partenza il giorno 0 e ritorno il giorno K-1=7 e che le disponibilità siano le seguenti (per ogni nome, tra parentesi si indicano il giorno iniziale e il giorno finale della disponibilità).

Paperino (3,5)

Paperoga (0,2)

Battista (1,3)

Gastone (5,6)

Archimede (4,7)

In questo caso, a Zio Paperone basta coinvolgere Paperoga, Paperino e Archimede per assicurarsi che il deposito sia sempre sorvegliato, e se la cava con tre regali.

Sapendo il numero di giorni di viaggio, e le disponibilità di ognuno, il vostro compito è quello di aiutare Zio Paperone a calcolare il minimo numero di persone che servono ad assicurare una sorveglianza continua al deposito.

Dati di input

Il file di input è costituito da $2+N$ righe. La prima riga contiene un intero positivo K , ovvero il numero di giorni del viaggio. La seconda riga contiene un intero positivo N , il numero di persone che hanno dato la disponibilità a Zio Paperone. Le restanti N righe contengono una coppia di interi A e B per ognuna delle N persone: questa coppia di interi rappresenta l'inizio e la fine della disponibilità della i -esima persona.

Dati di output

Il file di output deve contenere un solo intero positivo R , che è il numero minimo di persone necessarie ad assicurare una sorveglianza continua al deposito.

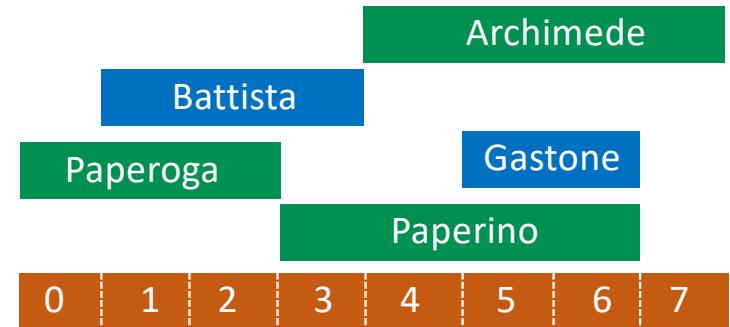
Assunzioni

$1 \leq K, N \leq 50$ Per ognuna delle N righe, si ha $0 \leq A \leq B \leq K-1$ Esiste sempre almeno una soluzione in ognuno dei casi di input.

Input e output: un esempio

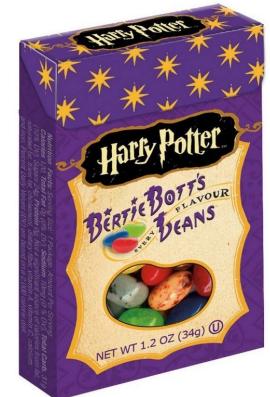
Esempi di input/output

File input.txt	File output.txt
8	
5	
3 5	
0 2	
1 3	
5 6	
4 7	



Un problema dalle Territoriali 2007: Giri sulla scopa Nimbus3000





Giri sulla Scopa Nimbus3000 (nimbus)

Difficoltà D = 2 (tempo limite 1 sec).

Descrizione del problema

Al celebre maghetto Harry Potter è stata regalata una scopa volante modello Nimbus3000 e tutti i suoi compagni del Grifondoro gli chiedono di poterla provare. Il buon Harry ha promesso che nei giorni a venire soddisferà le richieste di tutti, ma ogni ragazzo è impaziente e vuole provare la scopa il giorno stesso. Ognuno propone ad Harry un intervallo di tempo della giornata durante il quale, essendo libero da lezioni di magia, può fare un giro sulla scopa, e per convincerlo gli offre una fantastica caramella Tuttigusti+1. Tenendo presente che una sola persona alla volta può salire sulla Nimbus3000 in ogni istante di tempo, Harry decide di soddisfare, tra tutte le richieste dei ragazzi, quelle che gli procureranno la massima quantità di caramelle (che poi spartirà coi suoi amici Ron e Hermione). Aiutalo a trovare la migliore soluzione possibile.

Dati di input

Il file `input.txt` contiene nella prima riga un intero positivo N , che indica il numero di richieste, che sono numerate da 1 a N . Ognuna delle successive N righe contiene una coppia di interi. Ciascuna di tali righe contiene una coppia di interi positivi A e B , separati da uno spazio, a rappresentare la richiesta di poter utilizzare la scopa dall'istante iniziale A fino all'istante finale B , in cambio di una caramella (dove $A < B$). A tal fine, il tempo è diviso in istanti discreti numerati a partire da 1 in poi.

Dati di output

Il file `output.txt` è composto da una riga contenente un solo intero, che rappresenta il massimo numero di caramelle che Harry può ottenere.

Assunzioni

$1 < N < 1000$ Gli interi nelle N coppie sono distinti l'uno dall'altro (non esistono due interi uguali, anche in coppie diverse).

Input e output: un esempio

input.txt

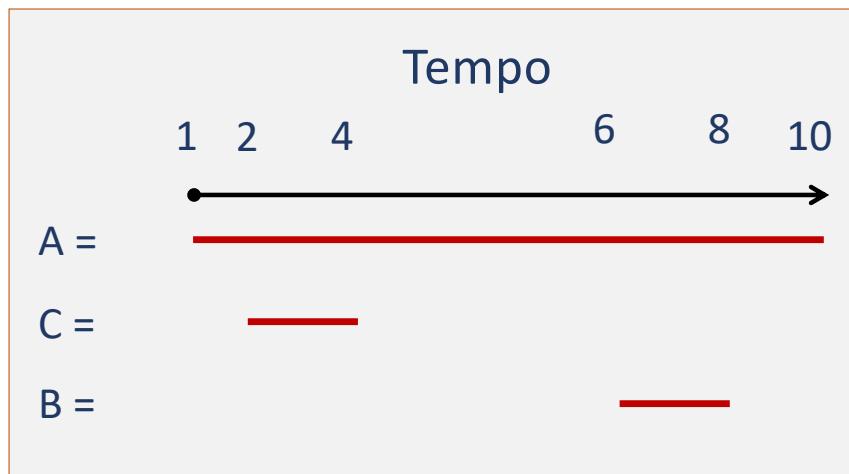
5
1 5
3 7
9 11
10 12
6 13

output.txt

2

Strategie a confronto (1/2)

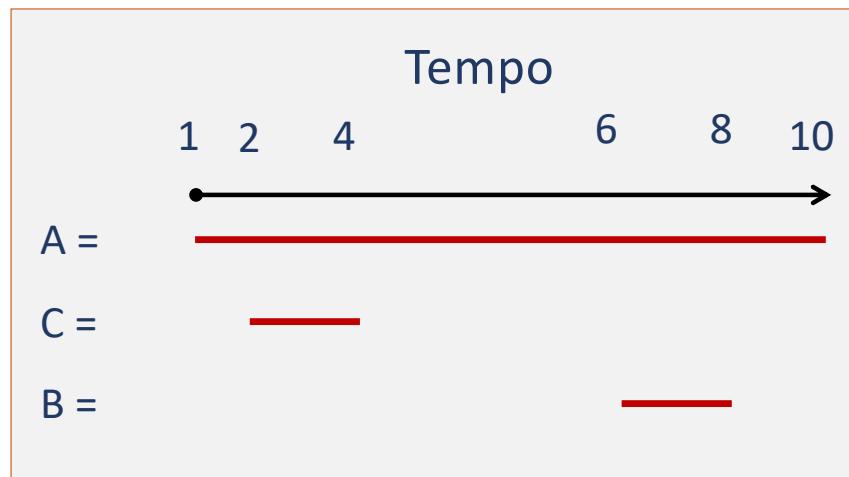
- Intervalli richiesti: $A = [1,10]$ $B = [6,8]$ $C = [2,4]$



- Soluzione 1: ordinamento per tempo di **inizio**
- Harry guadagna 1 caramella (intervallo A)

Strategie a confronto (2/2)

- Intervalli richiesti: $A = [1,10]$ $B = [6,8]$ $C = [2,4]$



- Soluzione 2: ordinamento per tempo di **fine**
- Harry guadagna 2 caramelle (intervalli B e C)

Codici di Huffman (e problemi simili)

Slide a cura di Nicola Prezza

Codici prefix-free

Un problema di compressione dati:

- Supponete di avere una stringa $S = \text{banana}$
- Vogliamo assegnare codici binari alle lettere in modo tale che:
 - Ogni codice non sia il prefisso di un altro (così possiamo concatenarli e decodificare)
 - La lunghezza in bit finale sia minima

Codici prefix-free

Un problema di compressione dati:

- Supponete di avere una stringa $S = \text{banana}$
- Vogliamo assegnare codici binari alle lettere in modo tale che:
 - Ogni codice non sia il prefisso di un altro (così possiamo concatenarli e decodificare)
 - La lunghezza in bit finale sia minima
- Esempio (non ottimale):
 - $b \rightarrow 0$, $a \rightarrow 10$, $n \rightarrow 11$
 - Stringa codificata: 01011101110
 - 11 bits. Possiamo fare di meglio?

Codici di Huffman

La soluzione ottima è **greedy!** Huffman, 1952. Lettere meno frequenti → codici più lunghi

Codici di Huffman

La soluzione ottima è **greedy!** Huffman, 1952. Lettere meno frequenti → codici più lunghi

1. Scriviamo la frequenza a fianco di ogni lettera

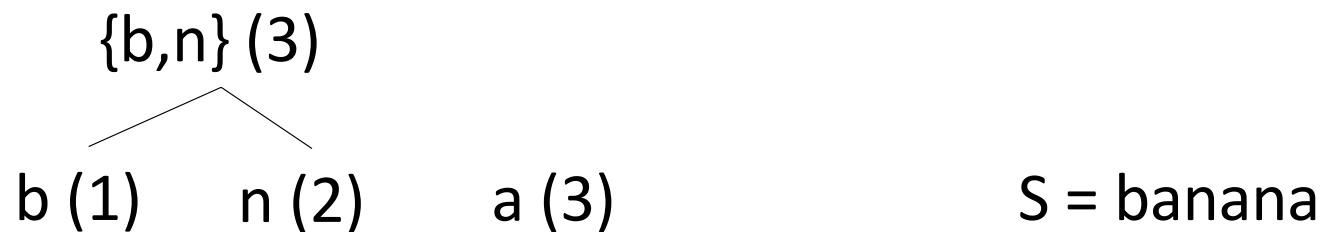
b (1) n (2) a (3)

S = banana

Codici di Huffman

La soluzione ottima è **greedy!** Huffman, 1952. Lettere meno frequenti → codici più lunghi

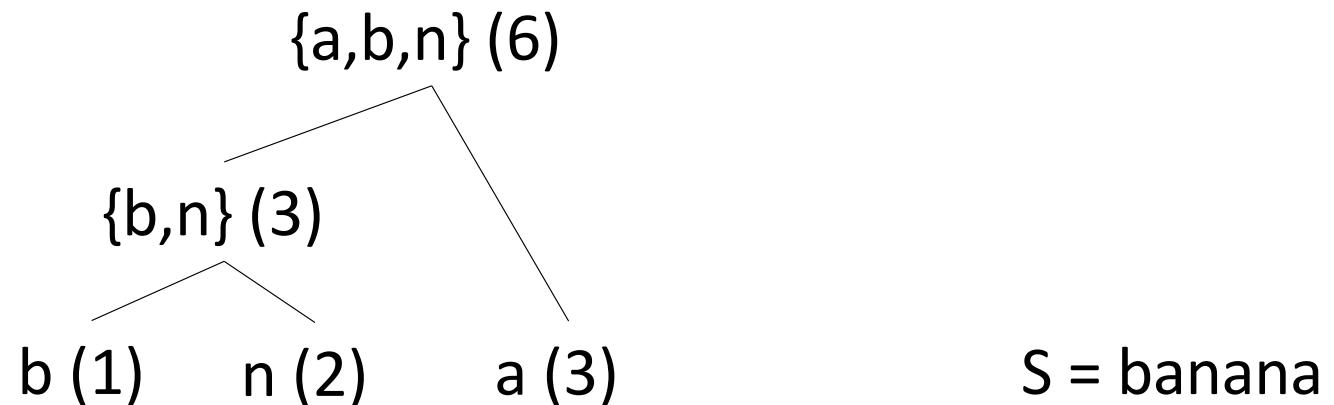
1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo



Codici di Huffman

La soluzione ottima è **greedy!** Huffman, 1952. Lettere meno frequenti → codici più lunghi

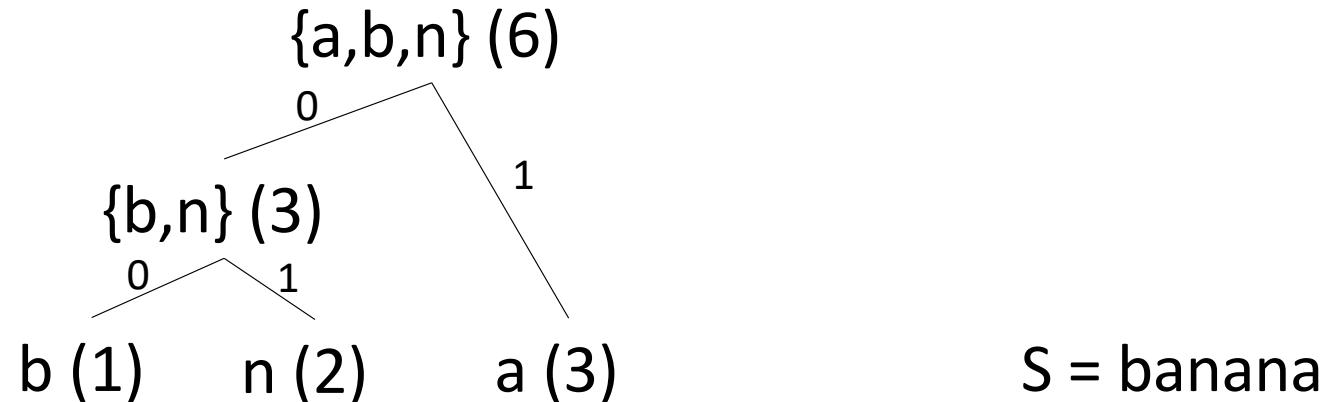
1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo
3. Ripetiamo trattando i gruppi come lettere



Codici di Huffman

La soluzione ottima è **greedy!** Huffman, 1952. Lettere meno frequenti → codici più lunghi

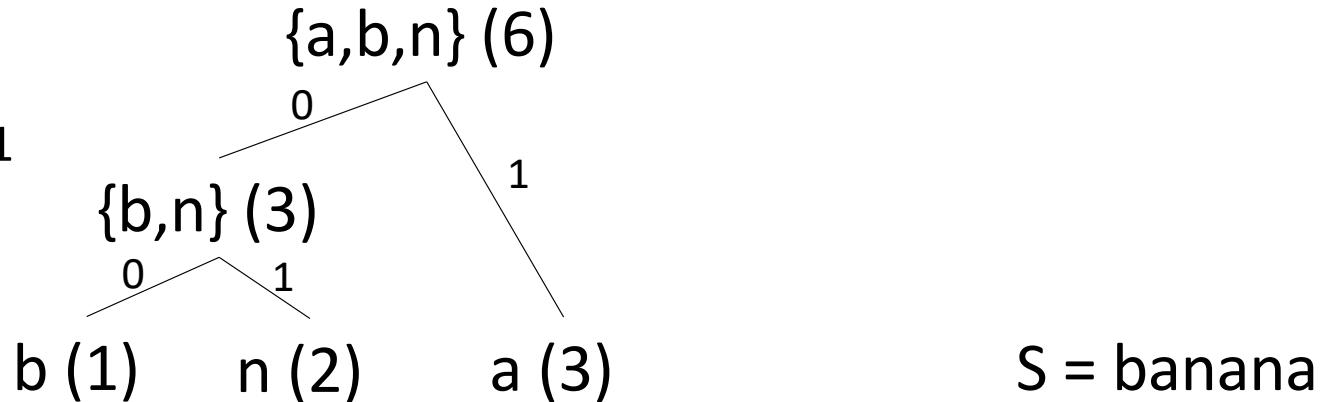
1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo
3. Ripetiamo trattando i gruppi come lettere
4. Sinistra: 0, destra: 1



Codici di Huffman

La soluzione ottima è **greedy!** **Huffman, 1952.** Lettere meno frequenti → codici più lunghi

1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo
3. Ripetiamo trattando i gruppi come lettere
4. Sinistra: 0, destra: 1
5. $b \rightarrow 00, a \rightarrow 1, n \rightarrow 01$



Codici di Huffman

Codice di Huffman:

- $b \rightarrow 00$, $a \rightarrow 1$, $n \rightarrow 01$
- Stringa codificata: 001011011
- 9 bit

Si dimostra che questo è ottimale!

Per i curiosi: <https://www.cs.toronto.edu/~radford/csc310.S02/week3b.pdf>)

Bonifici (Luiss contest 2020)

Slide a cura di Carlo Malagnino!



Descrizione del problema

La LuissAlgoCorp™ è una società peer-to-peer (senza un'autorità centrale) composta da N dipendenti. La società trae profitto dalla vendita di programmi per la risoluzione di problemi delle gare di programmazione. I dipendenti lavorano in autonomo: ognuno di essi sceglie un problema dalla lista dei problemi aperti posti dai clienti, lo risolve con un'implementazione efficiente in C++, invia la soluzione al cliente e riceve un compenso proporzionale alla difficoltà del problema sul proprio conto corrente.

Alla fine del mese, il profitto accumulato, indicato con p_i per l' i -esimo dipendente (per $i=1, \dots, N$), deve essere trasferito nel conto di un unico dipendente (uno qualunque), il quale avrà il compito di calcolare gli stipendi, bonus e promozioni di tutti i dipendenti in base alla loro produttività. Purtroppo, la banca richiede una commissione variabile per i bonifici! Se un dipendente invia x euro verso un altro dipendente il cui conto corrente contiene y euro, la commissione è pari all' 1% di $x + y$. Trovate la sequenza di bonifici tra dipendenti che porta al **minor costo totale** delle commissioni, tenendo presente che:

- All'inizio, il conto dell' i -esimo dipendente contiene p_i euro.
- Alla fine della sequenza di bonifici, l'intera cifra $p_1 + p_2 + \dots + p_N$ deve essere trasferita sul conto di un unico dipendente (uno qualunque).
- I bonifici vanno effettuati in modo isolato tra due conti correnti. In altre parole: non è ammesso che più dipendenti inviano denaro in contemporanea verso un conto corrente.

Dati di input

La prima riga del file di input contiene un intero T , il numero di casi di test. Seguono T casi di test, numerati da 1 a T . Ogni caso di test è preceduto da una riga vuota.

Ciascun caso di test è composto da $N + 1$ righe: la prima riga contiene il numero N di dipendenti, le successive N righe contengono i profitti p_1, p_2, \dots, p_N .

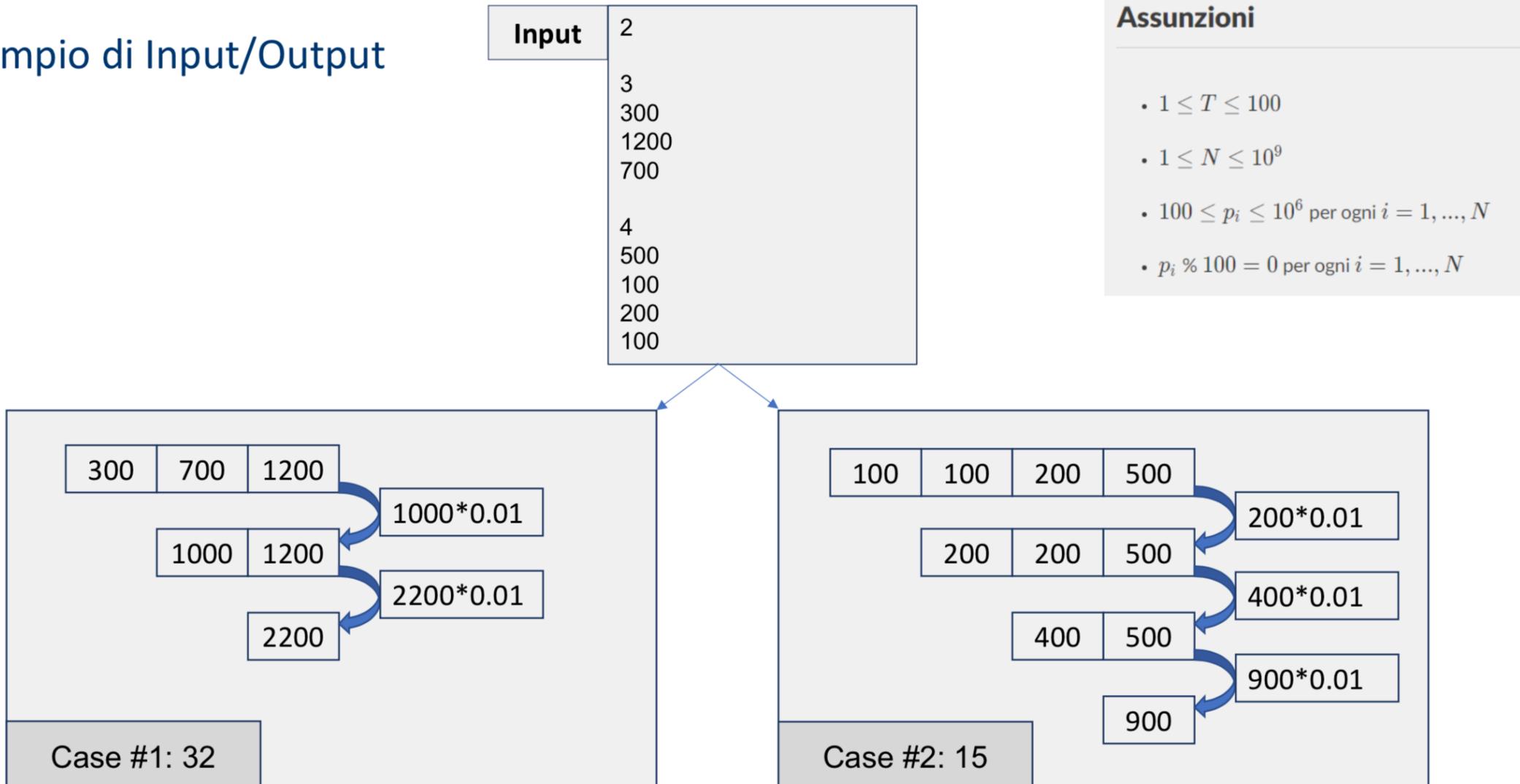
Dati di output

Il file di output deve contenere la risposta ai casi di test che sei riuscito a risolvere. Per ogni caso di test che hai risolto, il file di output deve contenere una riga con la dicitura

```
Case #t: s
```

dove t è il numero del caso di test (a partire da 1) e il valore s è un intero, il minimo costo totale delle commissioni.

Esempio di Input/Output



Soluzione in C++

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    int T;
    cin >> T;

    for(int t=1;t<=T;t++) {
        priority_queue<long long int>q;
        long long int N;
        cin >> N;

        for(long long int i=0;i<N;i++) {
            long long int c;
            cin >> c;
            q.push(-(c/100));
        }

        long long int ris=0;
        while(q.size()>1){
            long long int A,B;
            A=-q.top();
            q.pop();
            B=-q.top();
            q.pop();
            ris+=A+B;
            q.push(-(A+B));
        }
        q.pop();
        cout << "Case #" << t << ":" << ris << "\n";
    }
}
```

Sbarramento tattico (Territoriali 2010)

<https://training.olinfo.it/#/task/sbarramento/statement>



Sbarramento tattico (sbarramento)

Difficoltà D = 2.

Descrizione del problema

L'esercito di Orchi dell'Oscuro Signore degli Anelli marcia a ranghi serrati verso il Fosso di Helm. Per contrastarne la marcia, Re Theoden decide di richiamare tutte le sue N armate per creare uno sbarramento unico, con le seguenti regole.

- Campo di battaglia: è rappresentato da una tabella di dimensione $N \times N$, le cui righe e colonne sono numerate da 1 a N .
- Posizione: ognuna delle N armate occupa una posizione distinta $[i, j]$ nella tabella, all'incrocio tra la riga i e la colonna j .
- Movimento: permette di passare dalla posizione corrente $[i, j]$ a una vicina con un giorno di marcia: nord $[i - 1, j]$ (se $i > 1$), sud $[i + 1, j]$ (se $i < N$), est $[i, j + 1]$ (se $j < N$) e ovest $[i, j - 1]$ (se $j > 1$). Una sola armata alla volta si sposta con un movimento.
- Sbarramento: si crea ponendo tutte le armate su un'unica riga R della tabella, attraverso una serie di movimenti.

Theoden vuole calcolare il numero minimo di movimenti necessari per spostare tutte le armate in un unico sbarramento sulla riga R . Aiutate Theoden a calcolare tale numero minimo.

Dati di input

Il file `input.txt` è composto da $N + 1$ righe. La prima riga contiene due interi positivi N e R , separati da uno spazio: il numero N di righe e di colonne nella tabella (nonché il numero di armate) e l'indice R della riga su cui far convergere lo sbarramento delle armate. Ciascuna delle successive N righe contiene una coppia di interi i e j , separati da uno spazio, a indicare che un'armata è presente nella posizione $[i, j]$ della tabella.

Dati di output

Il file `output.txt` è composto da una sola riga contenente un intero non negativo, il minimo numero di movimenti per posizionare tutte le armate sulla riga R della tabella, in posizioni distinte all'interno di tale riga.

Assunzioni

- $2 \leq N \leq 500$.
- Durante un movimento, due o più armate non possono mai occupare la stessa posizione intermedia.

Esempi di input/output

File input.txt	File output.txt
8 3 5 5 1 6 2 2 6 5 3 2 7 1 1 2 8 1	31
8 5 5 7 5 2 5 3 5 6 5 1 5 8 5 5 5 4	0

Esercizi per casa



Esercizi

- Crittografia LWF (Territoriali 2017)

<https://training.olinfo.it/#/task/lwf/statement>

- Pausa caffè (OIS 2016)

https://training.olinfo.it/#/task/ois_caffe/statement

- N-restaurants (OIS 2019)

https://training.olinfo.it/#/task/ois_restaurants/statement

- Per un pugno di baht (Nazionali 2011)

<https://training.olinfo.it/#/task/baht/statement>