

Corso di preparazione per la selezione territoriale delle Olimpiadi di Informatica

Lezione 3. Algoritmi greedy

Luigi Laura

13 marzo 2023

LUISS 



Lezioni

Puoi accedere alle lezioni tramite YouTube o Webex:

 **YouTube** (preferibile se hai problemi di connessione)

 **Cisco webex** (puoi diventare ***Chad (Stacy) per un giorno*** condividendo la tua soluzione ai problemi che consideriamo di volta in volta!)



Programma di oggi

1. Soluzione esercizi della scorsa settimana
2. Introduzione alla tecnica Greedy
3. Tecnica Greedy: problemi di sequenziamento e problemi su intervalli
4. Un problema dalla finale OII 2022: Episodio I: un acquisto difficile
5. Tecnica Greedy: problemi di compressione
6. Problemi: Pulizie di Primavera (Finale OII 2021), Finanza Creativa (OIS 2014)
7. Compiti da fare a casa (per la prossima volta)

Soluzioni degli esercizi della scorsa settimana



Compiti a casa della scorsa settimana

- Tecnico pazzo: [pazzo](#)
- Paletta: [oii_paletta](#)
- Regali: [ois_regali](#)
- Seats: [ois_seats](#)
- Maxim: [ois_maxim](#)

Tutti vogliamo
diventare chad
per un giorno



Se sei collegato/a tramite
Webex alza la mano
(pulsante  vicino al tuo
nome nella lista partecipanti)
per presentare la tua
soluzione ai problemi della
scorsa settimana

La tecnica greedy



Slide a cura di Irene Finocchi

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  $S$ 
```

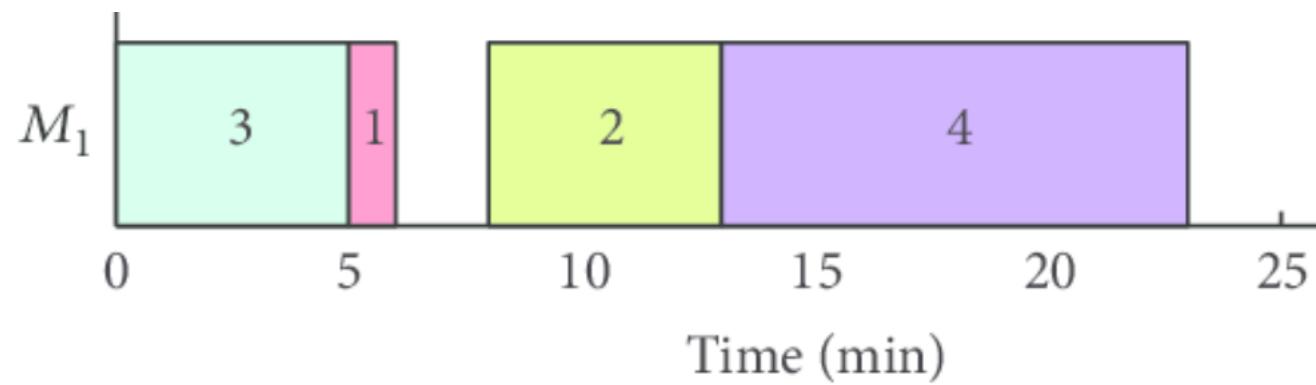
Perché goloso?

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

Somme Costose

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



GATOR – Gara Allenamento TOR Vergata 2016

Gara online, 9-10 aprile 2016

somme • IT

Somme costose (somme)

Gabriele sta studiando il seguente problema. Deve sommare dei numeri tra di loro, ma ogni volta può sommare solo una coppia di numeri, e il **costo** di questa operazione è pari al risultato della somma.

Ad esempio, volendo sommare 1, 2 e 3, ci sono tre possibilità:

1. Sommiamo $(1 + 2) + 3 = 3 + 3 = 6$: la prima operazione $(1 + 2)$ ci costa 3, la seconda $(3 + 3)$ ci costa 6, il totale per fare questa sequenza di somme è quindi 9.
2. Sommiamo $(1 + 3) + 2 = 4 + 2 = 6$: la prima operazione $(1 + 3)$ ci costa 4, la seconda $(4 + 2)$ ci costa 6, il totale per fare questa sequenza di somme è quindi 10.
3. Sommiamo $(2 + 3) + 1 = 5 + 1 = 6$: la prima operazione $(2 + 3)$ ci costa 5, la seconda $(5 + 1)$ ci costa 6, il totale per fare questa sequenza di somme è quindi 11.

Il vostro compito, dato un insieme di numeri da sommare, è quello di trovare il costo minimo per sommarli tutti tra di loro. Ad esempio, come visto sopra, se l'insieme dei numeri da sommare è 1, 2 e 3, il costo minimo per sommarli è 9.

Dati di input

Il file `input.txt` contiene due righe. La prima riga contiene l'intero N , il numero di elementi da sommare. La seconda riga contiene N interi a_1, \dots, a_N separati da uno spazio, corrispondenti alla lista dei numeri da sommare tra loro.

Dati di output

Sul file `output.txt` stampare il costo minimo necessario per sommare tra di loro tutti i numeri.

Assunzioni

- $2 \leq N \leq 100\,000$.
- $1 \leq a_i \leq 1000$ per ogni $i = 1, \dots, N$.

Esempi di input/output

input.txt	output.txt
3 2 1 3	9
5 10 7 2 9 3	67

MATH HALLOWEEN COSTUME IDEA #1



AT EACH STAGE, STEAL THE CANDY FROM A KID WITH CANDY NEAREST TO THE ONE YOU JUST STOLE CANDY FROM.



Esempio: greedy KO

Un problema di cambio di denaro



Il problema e l'algoritmo greedy "naturale"

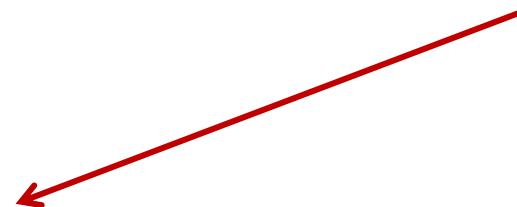
- Input: un numero intero positivo n
- Output: il più piccolo numero intero di banconote per cambiare n Euro usando pezzi da 20, 10, 5 e 1 Euro
- Esempi
 - $n = 58$, 7 banconote: $20+20+10+5+1+1+1$
 - $n = 18$, 5 banconote: $10+5+1+1+1$
- Algoritmo greedy
 - Dispensa una banconota alla volta
 - Ad ogni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



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

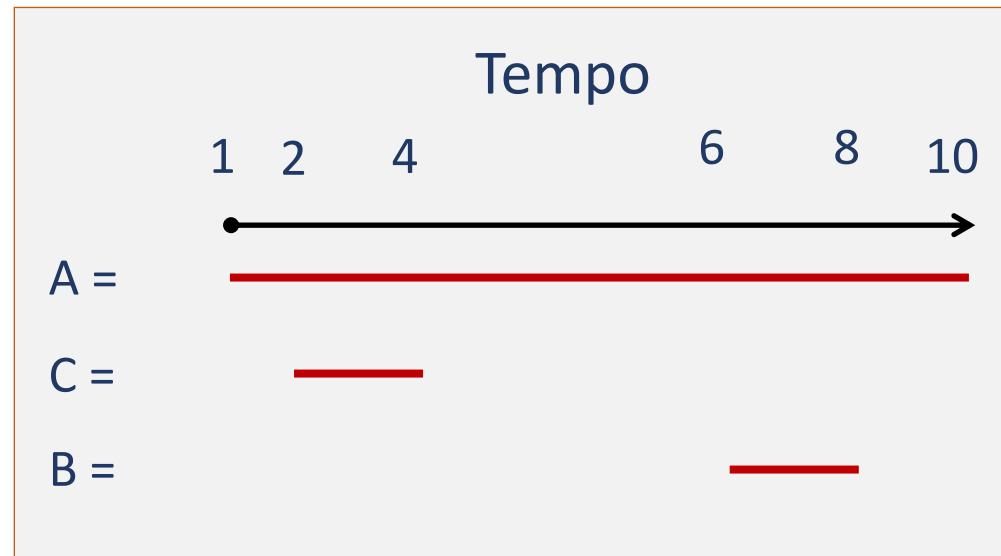
6 13

output.txt

2

Strategie a confronto (1/2)

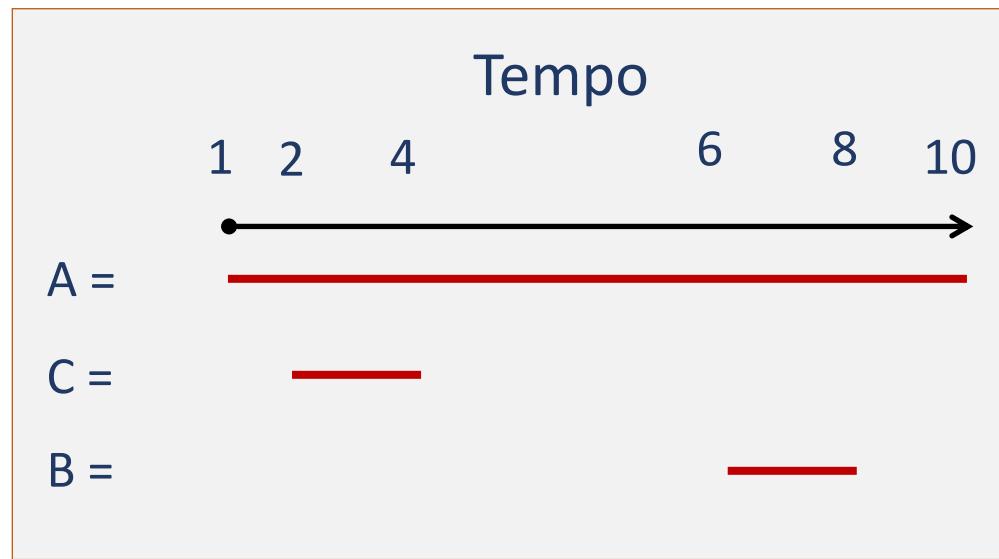
- Intervalli richiesti: $A = [1,10]$ $B = [6,8]$ $C = [2,4]$



- 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)

Ottimalità: intuizione

Se scegliamo l'intervallo compatibile con le scelte precedenti e con istante di fine minore in seguito **non troveremo un intervallo che abbiamo scartato e che sarebbe stato meglio scegliere**. Perché?

1. Nella migliore delle ipotesi **si sostituirebbe all'intervallo precedente non migliorando il numero di caramelle**, ma potenzialmente peggiorando l'istante di fine e quindi potenzialmente peggiorando il numero di intervalli che potranno essere presi in futuro
2. Nella peggiore delle ipotesi **porterebbe ad eliminare intervalli presi in precedenza** quindi potenzialmente peggiorando il numero di caramelle

Implementazione

```
1 struct Intervallo
2 {
3     int inizio, fine;
4 };
5
6 bool compare_intervalli(const Intervallo &a, const Intervallo &b)
7 {
8     if (a.fine < b.fine)
9         return true;
10    else
11        return false;
12 }
13
14 int N;
15 Intervallo giri[1000];
```

```
17 int main()
18 {
19     ifstream in("input.txt");
20     ofstream out("output.txt");
21     in >> N;
22     for (int i=0; i<N; i++)
23     {
24         in >> giri[i].inizio >> giri[i].fine;
25     }
26     sort(giri,giri+N,compare_intervalli);
27     int fine_attuale = giri[0].fine;
28     int caramelle = 1;
29     for (int i=1; i<N; i++)
30     {
31         if (fine_attuale < giri[i].inizio)
32         {
33             fine_attuale = giri[i].fine;
34             caramelle++;
35         }
36     }
37     out << caramelle << endl ;
38 }
39 }
```

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.

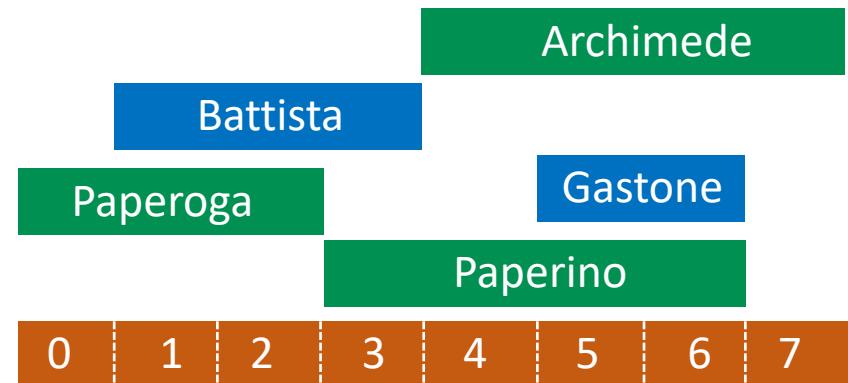
Assunzioni

$1 \leq K, N \leq 50$ 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	



Idea risolutiva

- Ordinamento per data di inizio e, a parità di data di inizio, per lunghezza crescente dell'intervallo
- Tutti i K giorni devono essere “coperti”
- Esiste almeno un intervallo che inizia dal giorno 0
 - Se è uno solo, lo scelgo (non ho altre possibilità)
 - Se è più di uno, scelgo quello con la data di fine maggiore (se ne scegliessi un altro coprirei meno giorni e non avrei alcun vantaggio)
- Continuo in questo modo sui giorni rimanenti, considerando come prossimo giorno da coprire quello successivo alla data di fine dell'ultimo intervallo scelto

Implementazione

```
1 class Turno {
2     public:
3         int inizio, fine;
4         bool operator< (const Turno& t) const {
5             return inizio < t.inizio;
6         };
7         Turno turni[50];
8         int K,N;
```

```
11    int main() {
12        ifstream in("input.txt");
13        ofstream out("output.txt");
14        in >> K >> N;
15        for (int i=0; i<N; i++)
16            in >> turni[i].inizio >> turni[i].fine;
17        sort(turni, turni + N);
18        int fine = -1;
19        int i = 0;
20        int quanti = 0;
21        while (fine < K-1){
22            int max = turni[i].fine;
23            while (turni[i].inizio - 1 <= fine){
24                if (turni[i].fine > max) max = turni[i].fine;
25                i++;
26            }
27            fine = max;
28            quanti++;
29        }
30        out << quanti << endl;
31        return 0;
32    }
```

Attenzione! Non tutti i problemi su
intervalli si risolvono in maniera
greedy...

Missioni Segrete
<https://training.olinfo.it/#/task/missioni/statement>

Il Commissario Basettoni ha presentato a Topolino le missioni che egli dovrà svolgere segretamente nel corso dell'anno. Per ogni missione, oltre al luogo da raggiungere, Basettoni ne indica la durata in giorni e la data massima entro cui deve essere completata. In altri termini, la missione può iniziare in qualunque giorno dell'anno ma deve durare esattamente il numero di giorni indicato e terminare non oltre la data di scadenza.

Topolino, presa la lista delle missioni ricevuta da Basettoni, ordina tali missioni in base alla loro data di scadenza. Quindi, numera i giorni dell'anno da 1 a 365 (non esistono anni bisestili a Topolinia) e trasforma le date di scadenza in numeri secondo tale numerazione. Per esempio, se una missione dura 15 giorni e deve essere svolta entro il 18 febbraio, Topolino la vede semplicemente come una coppia di interi 15 49 (in quanto il 18 febbraio è il quarantanovesimo giorno dell'anno).

Poiché può svolgere una sola missione alla volta, Topolino sa che potrebbe svolgerne solo alcune pur iniziando una missione il giorno immediatamente successivo a quello in cui termina la precedente missione. Vuole perciò sapere il numero massimo di missioni che è in grado di svolgere rispettando i vincoli sulla loro durata e scadenza. Supponendo che Topolino già fornisca le coppie di interi ordinate per scadenza (il secondo membro delle coppie), aiutatelo a calcolare il massimo numero di missioni che può svolgere.

Per esempio, se ci sono quattro missioni, una di tre giorni da terminare entro il 5 gennaio, una di quattro giorni entro l'8 gennaio, una di tre giorni entro il 9 gennaio e una di 6 giorni entro il 12 gennaio, Topolino vi fornisce la lista di quattro coppie 3 5, 4 8, 3 9 e 6 12. Il numero massimo di missioni che può svolgere è pari a tre, ossia le missioni corrispondenti alle coppie 3 5, 3 9 e 6 12: la prima missione inizia il primo di gennaio e termina il 3 gennaio; la seconda inizia il 4 gennaio e termina il 6 gennaio; la terza inizia il 7 gennaio e termina il 12 gennaio. (Notare che, scegliendo la missione corrispondente alla coppia 4 8, Topolino può svolgere al più due missioni.)

Dati di input

Il file `input.txt` è composto da $N+1$ righe. La prima riga contiene un intero positivo che rappresenta il numero N di missioni presentate da Basettoni a Topolino.

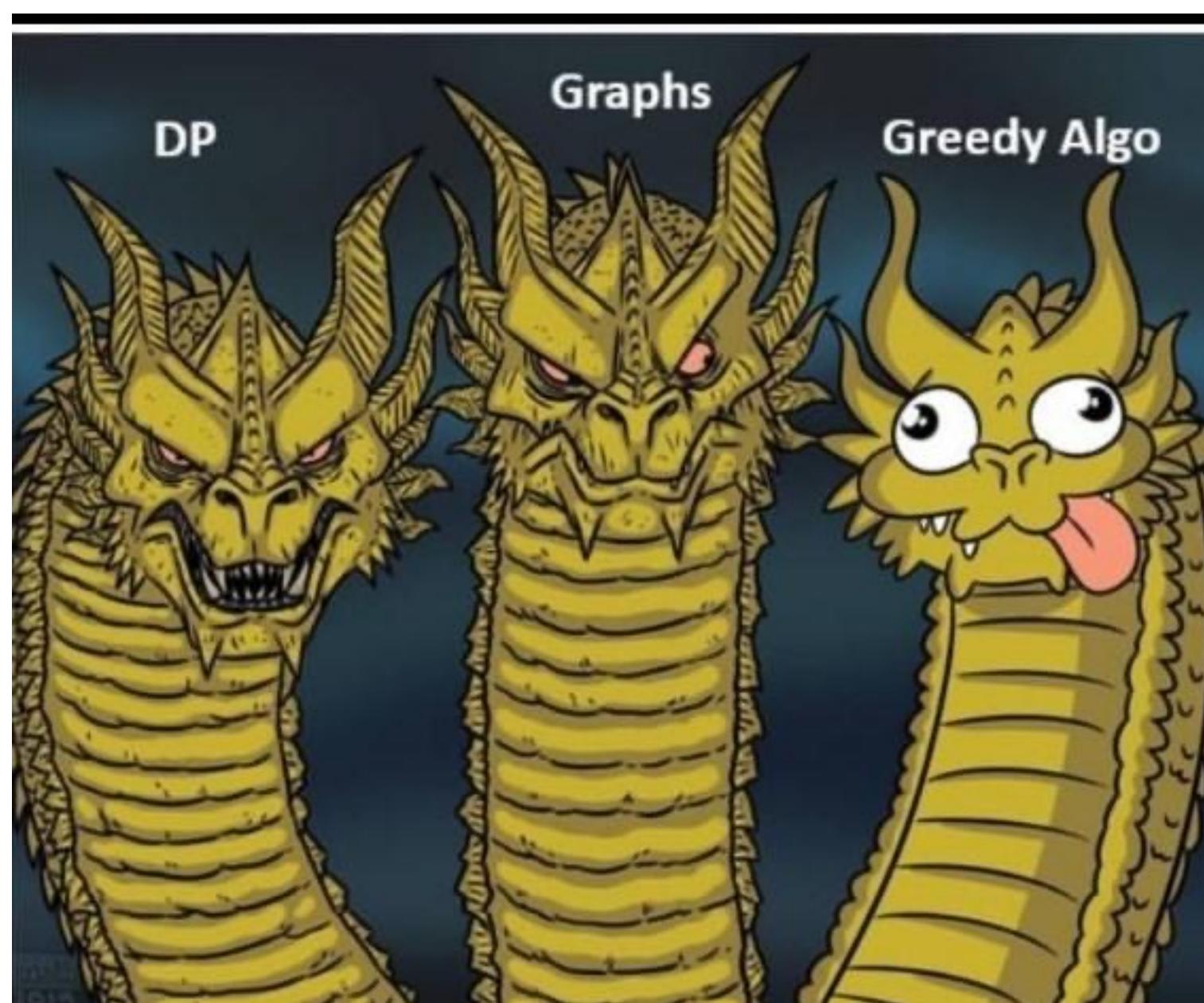
Le successive N righe rappresentano durata e scadenza delle missioni: ciascuna riga è composta da due interi d e s separati da uno spazio, a rappresentare che la corrispondente missione dura d giorni e deve essere completata entro l' s -esimo giorno dell'anno.

Dati di output

Il file `output.txt` è composto da una sola riga contenente un intero che rappresenta il massimo numero di missioni che Topolino può svolgere rispettando i vincoli su durata e scadenza.

Esempi di input/output

File input.txt	File output.txt
4 3 5 4 8 3 9 6 12	3



Non a caso, su codeforces...

Un problema dalla finale OII 2022: Episodio I: un acquisto difficile

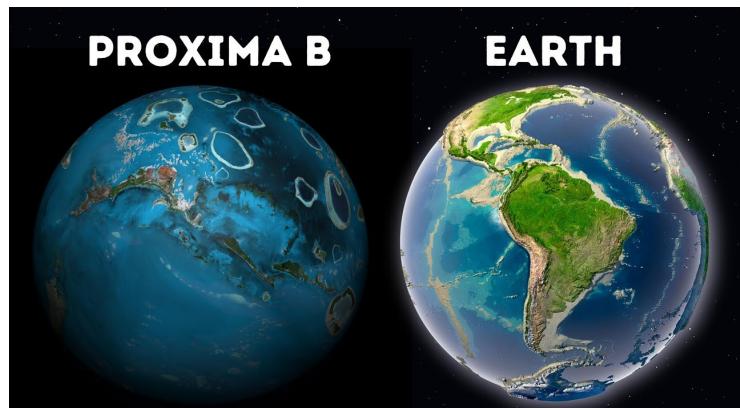
https://training.olinfo.it/#/task/oii_acquisti/statement

Episodio I: un acquisto difficile (acquisti)

Dopo lunghi secoli di osservazione del pianeta Terra, la direzione delle *Frazioni Unite* di Proxima B ha scoperto l'esistenza delle Olimpiadi Italiane di Informatica 2017 a Trento, e ha deciso di inviare una sua delegazione alla prossima edizione! Dopo una dura selezione, il giovane フィリップ・ヤマグチが
ha guadagnato il suo posto nella delegazione, e si sta ora preparando per il lungo viaggio verso la Terra, che durerà ben 150 anni di Proxima B, corrispondenti a poco meno di 5 anni solari.

In un viaggio come questo, non si può pensare di non portarsi dietro un po' di souvenir! フィリップが
si è fatto prendere un po' la mano, procurandosi souvenir di T tipologie diverse. Della tipologia i ($i = 0, \dots, T - 1$) ha acquistato S_i souvenir, ciascuno di identico peso pari a i qite.¹ Ora deve decidere quale valigia acquistare tra M modelli diversi, ciascuno con una portata massima di P_i qite ($i = 0, \dots, M - 1$).

Aiuta フィリップ a effettuare una scelta consapevole, calcolando *per ogni modello* di valigia quanti souvenir al massimo potrebbe contenere, tra quelli che ha già acquistato!



Esempi di input/output

stdin	stdout
5 5 3 2 7 8 6 9 54 1 100 40	8 23 4 26 20
10 5 0 3 2 7 8 6 95419852 14736461 0 2 1 6 5040 675674438 32786954198521	1 4 851 110156340 110156341

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 bit. 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

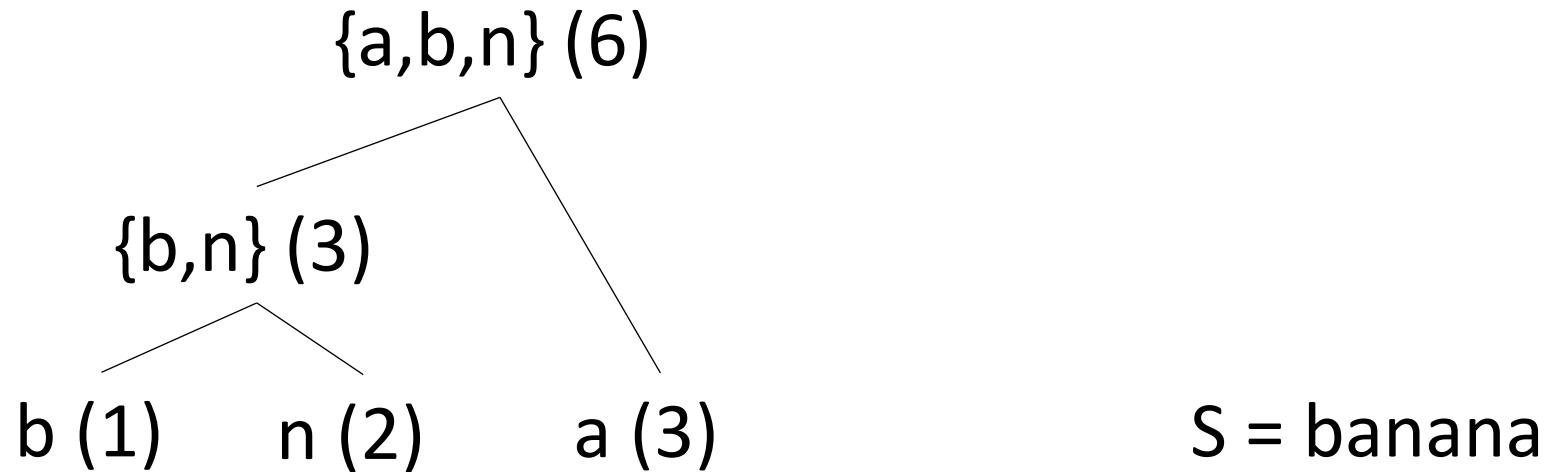
1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo



Codici di Huffman

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

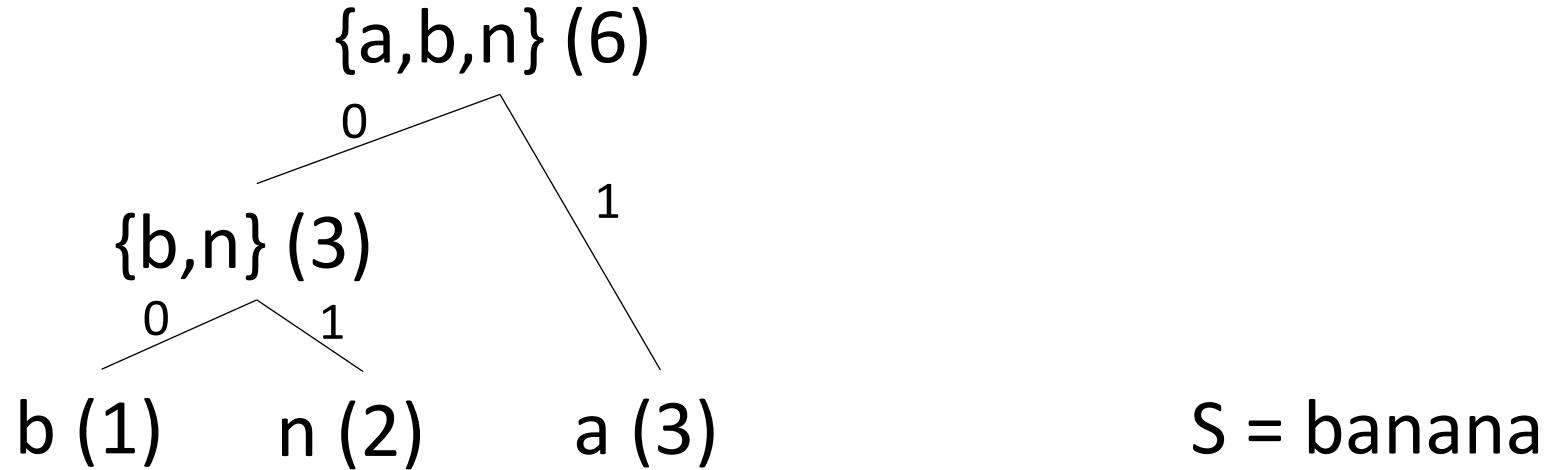
1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo
3. Ripetiamo trattando i gruppi come lettere



Codici di Huffman

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

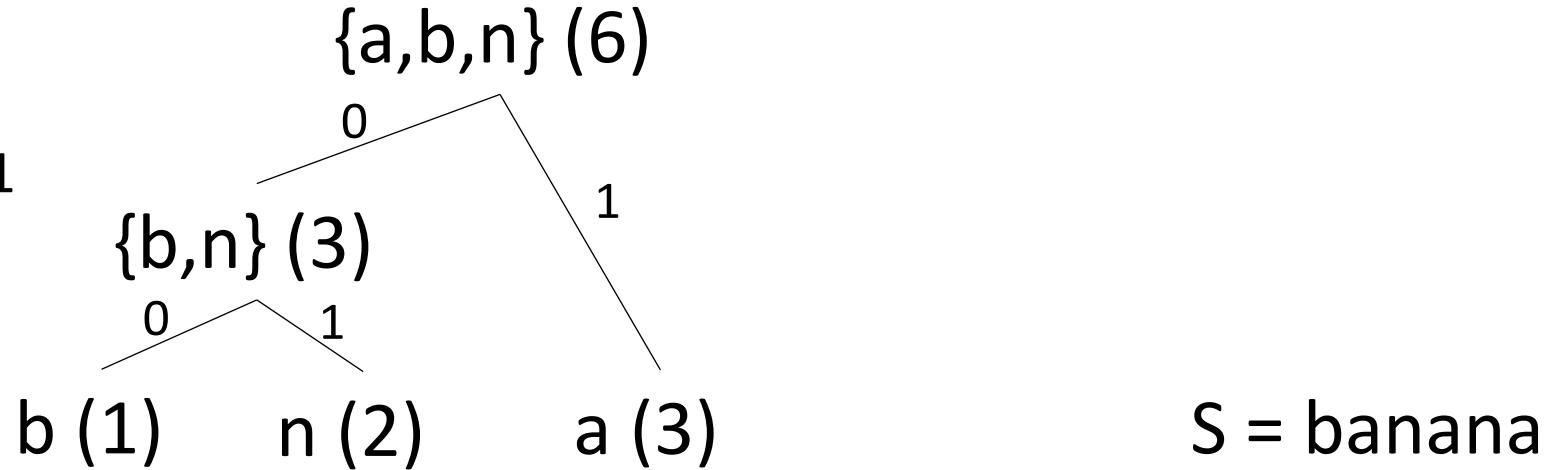
1. Scriviamo la frequenza a fianco di ogni lettera
2. Raggruppiamo le due lettere meno frequenti, aggiorniamo la frequenza del gruppo
3. Ripetiamo trattando i gruppi come lettere
4. Sinistra: 0, destra: 1



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 → 00, a → 1, n → 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>

Un problema dalla finale OII 2021: Pulizie di Primavera

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

Pulizie d'autunno (trendytrash)

Edoardo ha una stanza rettangolare piena di sacchi della spazzatura, di cui si vorrebbe liberare effettuando alcuni viaggi al centro di raccolta differenziata.

La stanza si può vedere come una griglia $N \times M$ (cioè con N righe, numerate da 0 a $N - 1$, ed M colonne, numerate da 0 a $M - 1$). Per ogni $0 \leq i < N, 0 \leq j < M$, vi è un sacco della spazzatura all'incrocio della riga i e colonna j (per un totale di $N \cdot M$ sacchi).

Per raccogliere i sacchi, Edoardo usa un carrello con il quale percorre una riga o una colonna della stanza, raccogliendo **tutti e soli** i sacchi presenti in quella riga o colonna tra quelli rimasti. Dopodiché porta tali sacchi al centro di riciclo, depositandoli nel contenitore opportuno.

Essendosi allenato duramente in palestra durante gli ultimi mesi, Edoardo può portare quanti sacchi vuole in un singolo viaggio. Tuttavia, dato che alcuni dei sacchi contengono carta e altri plastica, non vuole mai raccogliere sacchi di due tipi diversi insieme, altrimenti sarebbe troppo complicato poi separarli al centro di riciclo.

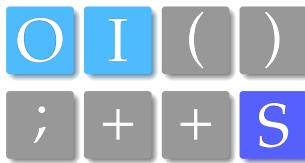
Edoardo vuole liberarsi del maggior numero possibile di sacchi della spazzatura. Aiutalo a determinare il minimo numero di sacchi che dovranno rimanere nella stanza al termine dei suoi viaggi.

Esempi di input/output

stdin	stdout
3 3 101 101 110	6
3 3 100 100 111	0
4 5 00010 01110 01000 11110	4

Un problema della OIS 2014: Finanza Creativa

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



Finanza creativa (bilancio)

Limite di tempo: 1.0 secondi

Limite di memoria: 256 MiB

La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, non accenna ad uscire dal periodo di crisi nonostante i forti e ben ponderati tagli al personale di recente effettuati. Per fortuna il *CEO* ha avuto una nuova geniale idea: affidarsi al massimo esperto mondiale in campo di finanza creativa. L'esperto ha già ricevuto il bilancio dal reparto contabilità, e la situazione è disastrosa: arrivati a questo punto l'unico modo che conosce per rassicurare gli azionisti e prendere tempo è quello di effettuare qualche piccolo ritocco ai libri contabili.

Più precisamente, vuole ricorrere al suo fedele bianchetto per correggere il totale U delle uscite. Inoltre, grazie agli insegnamenti di lunghi anni di esperienza, sa che per contenere i rischi dell'operazione non deve assolutamente eccedere le K cifre cancellate (sulle N complessive del totale U).

Aiuta l'esperto di finanza creativa a trovare il minimo intero ottenibile cancellando K cifre dall'intero U !

Esempi di input/output

input.txt	output.txt
5 2 1 9 5 0 3	1 0 3
input.txt	output.txt
9 3 9 4 1 5 7 1 1 2 3	1 5 1 1 2 3
input.txt	output.txt
10 5 1 3 2 0 2 1 8 5 3 6	0 1 5 3 6

Spiegazione

Nel **primo caso di esempio** conviene cancellare le due cifre più alte (5 e 9).

Nel **secondo caso di esempio** conviene cancellare le due cifre più significative (9 e 4), e la restante cifra più alta (il 7).

Nel **terzo caso di esempio** conviene cancellare le tre cifre più significative (1, 3 e 2), il 2 ancora successivo e la restante cifra più alta (l'8).

Compiti a casa (Esercizi Olimpiadi)



Compiti a casa

Prova a risolvere i seguenti problemi:

- [Rifiuti da riciclare](#)
- [Torta di Compleanno](#)
- [Aggiornamento della macchina virtuale](#)
- [Foglietto illustrativo](#)
- [Accampamento](#)

Accampamento si risolve con una greedy? o vi servono altri strumenti (DP)?