PROGETTO RICERCA OPERATIVA (HASH CODE 2020)

GIULIANO DI GIUSEPPE - M63001322
SOSSIO CIRILLO – M63001329
ANTONIO AVOLIO – M63001352

INDICE

INTRODUZIONE

Spiegazione del problema (1/2)

- Date (L) biblioteche, a ogni biblioteca è associato un insieme di libri.
- A ogni libro è associato un ID e un punteggio se viene scansionato.
- Prima di poter scansionare un libro la biblioteca deve registrarsi ed impiega un determinato numero di giorni per registrarsi, è possibile registrare una biblioteca alla volta.
- Inoltre dopo il giorno D non è più possibile scansionare libri.
- Il punteggio è la somma dei punteggi dei singoli libri scansionati, se un libro viene scansionato piu volte il suo punteggio viene calcolato solo una volta.

Spiegazione del problema (2/2)

Variabili generali

- ‘L’ = numero totale di biblioteche
- ‘B’ = numeri di libri presenti totali
- ‘D’= numero di giorni disponibili per scannerizzare il massimo dei libri

Ogni biblioteca ha i seguenti attributi:

- ‘registrazione’ = tempo impiegato per registrarsi
- ‘numero di libri massimi scannerizzati su giorno’ = numero di libri che puo scansionare ogni giorno
- ‘numero_di_libri’ = numero di libri presenti nella biblioteca

Ogni Libro ha i seguenti attributi:

- ‘ID’ = un ID che va da 0 a B-1
- ‘score’ = che indica il punteggio ottenuto quando viene scansionato il libri

Vincoli:

- TEMPORALI: Il programma inizia il giorno 0 e finisce il giorno ‘D’ , la registrazione puo essere fatta dal giorno 0,
- REGISTRAZIONE : è possibile registrare solo una biblioteca alla volta.
- SCORE: i libri gia scansionati danno un score uguale a 0
- DUPLICATI : se un libro viene scannerizzato piu volte il punteggio viene calcolato solo una volta

INPUT (1/2)

Prima riga:

- B L D : compresi tra 0 e 10^5

Seconda riga :

- S[0…L] : contiene lo score del libro i-esimo

Ogni 2 righe successive avremo :

- 1° riga : Nj Tj Mj : numero di libri presenti , tempo di registrazione , numero di libri scansionabili a giorno
- 2° riga : Idj : gli ID dei libri presenti nella libreria

INPUT (2/2)

- Esempio

Input file	Description
`6 2 7`	There are 6 books, 2 libraries, and 7 days for scanning.
`1 2 3 6 5 4`	The scores of the books are 1, 2, 3, 6, 5, 4 (in order).
`5 2 2`	Library 0 has 5 books, the signup process takes 2 days, and the library can ship 2 books per day.
`0 1 2 3 4`	The books in library 0 are: book 0, book 1, book 2, book 3, and book 4.
`4 3 1`	Library 1 has 4 books, the signup process takes 3 days, and the library can ship 1 book per day.
`3 2 5 0`	The books in library 1 are: book 3, book 2, book 5 and book 0.

OUTPUT (1/2)

- I file che deve essere consegnato dovra essere formattato nel seguente modo:
- 1° riga : numero di librerie registrate
- Ogni 2 righe successive avremo :
    - Prima riga : ID libreria , Numero di libri scansionati
    - Seconda riga: IDs libri scansionati

OUTPUT(2/2)

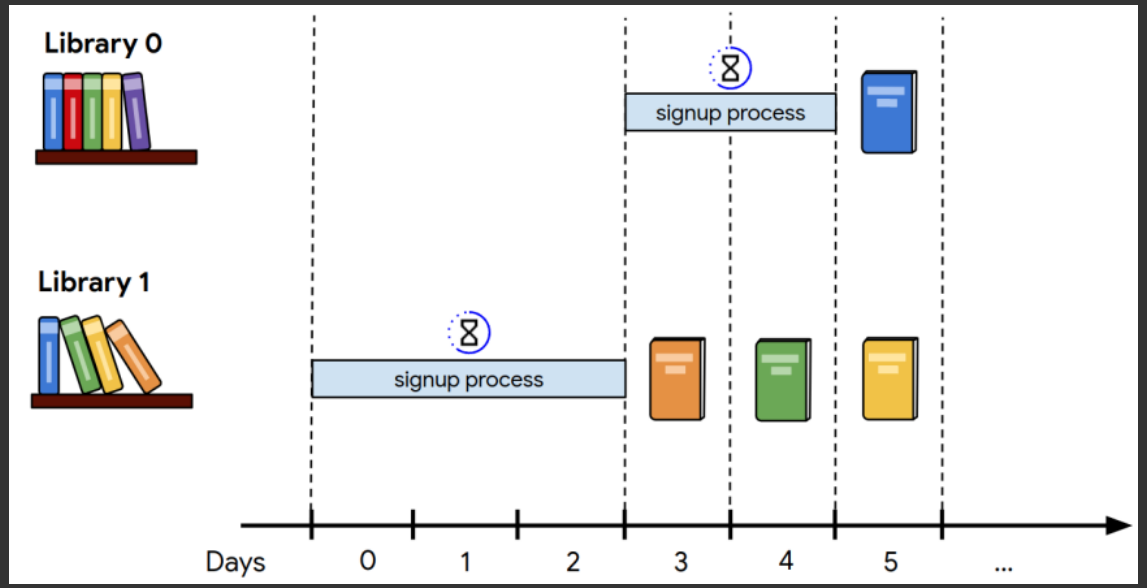
- Esempio :

Submission file	Description
2	Two libraries will be signed up for scanning.
1 3	The first library to do the signup process is library 1. After the signup process it will send 3 books for scanning.
5 2 3	Library 1 will send book 5, book 2, and book 3 in order.
0 5	The second library to do the signup process is library 0. After the signup process it will send 5 books.
0 1 2 3 4	Library 0 will send book 0, book 1, book 2, book 3 and book 4 in that order.

Esempio (1/2)

- La libreria 0 impiega 2 giorni per registrasi
- La libreria 1 impiega 3 giorni per registrarsi
- La libreria 1 si registra prima della libreria 2

- Quindi la libreria 1 si registra il giorno 0 e il giorno 3 puo iniziare a scannerizzare i libri , mentre la libreria 0 inizia il giorno 3 a registrarsi ma puo scannerizzare dal giorno 5



Esempio «a_example» (2/3)

- A sinistra è possibile vedere un possibile input e a destra una possibile soluzione

Input file	Description
`6 2 7`	There are 6 books, 2 libraries, and 7 days for scanning.
`1 2 3 6 5 4`	The scores of the books are 1, 2, 3, 6, 5, 4 (in order).
`5 2 2`	Library 0 has 5 books, the signup process takes 2 days, and the library can ship 2 books per day.
`0 1 2 3 4`	The books in library 0 are: book 0, book 1, book 2, book 3, and book 4.
`4 3 1`	Library 1 has 4 books, the signup process takes 3 days, and the library can ship 1 book per day.
`3 2 5 0`	The books in library 1 are: book 3, book 2, book 5 and book 0.

Figura 1 : input

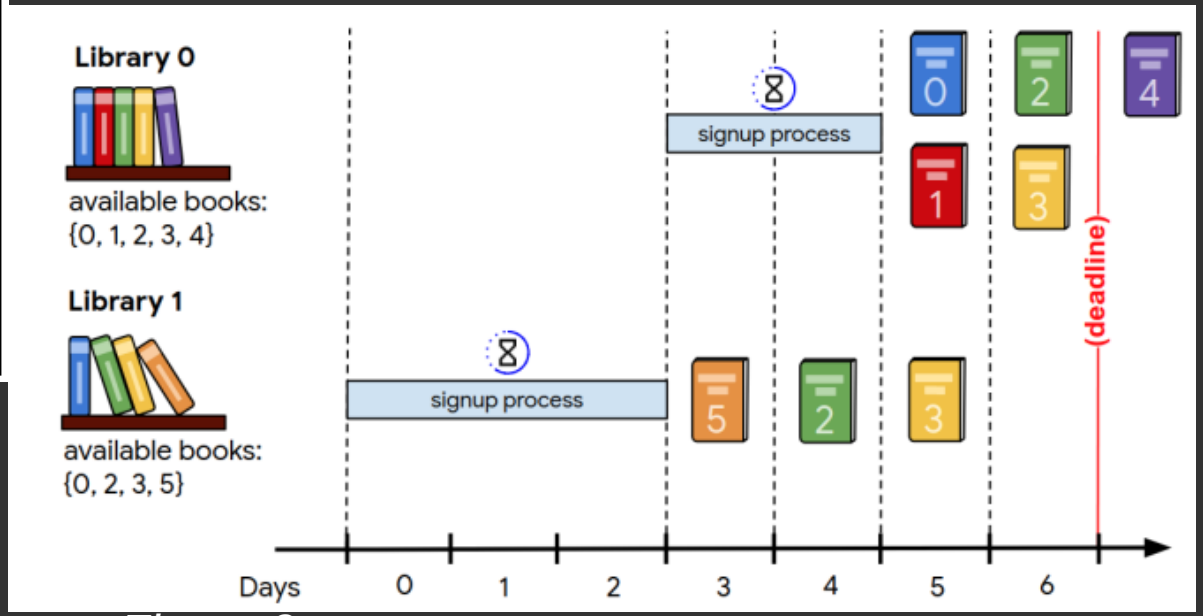


Figura 2 : Rappresentazione possibile soluzione

Esempio (3/3)

- Alla fine, vengono scansionati il libro 0, il libro 1, il libro 2, il libro 3 e il libro 5. I loro punteggi sono rispettivamente 1, 2, 3, 6 e 4. Il che si traduce in un punteggio totale di 16. Nota che il libro 2 e il libro 3 vengono scansionati due volte, ma i loro punteggi vengono comunque contati solo una volta. Inoltre, nota che non viene assegnato alcun punteggio per il libro 4, poiché viene scansionato il giorno 6.
- Se l'algoritmo scansionasse il giorno 6 il libro 4 piuttosto che il libro 2 o 3 lo score diventerebbe ottimo con valore pari a 21

CASI DI TEST

A - example
Books: **6** Libraries: **2** Time: **7**
The signup of all libraries takes **5** days, which is **71.43%** of all 7 days.

B - read on
Books: **100,000** Libraries: **100** Time: **1,000**
The signup of all libraries takes **1,181** days, which is **118.10%** of all 1,000 days.

C - incunabula
Books: **100,000** Libraries: **10,000** Time: **100,000**
The signup of all libraries takes **5,060,797** days, which is **5,060.80%** of all 100,000 days.

E - so many books
Books: **100,000** Libraries: **1,000** Time: **200**
The signup of all libraries takes **5,432** days, which is **2,716.00%** of all 200 days.

F - libraries of the world
Books: **100,000** Libraries: **1,000** Time: **700**
The signup of all libraries takes **166,192** days, which is **23,741.71%** of all 700 days.

STRUTTURA GENERALE DEGLI ALGORITMI IMPLEMENTATI

1. Lettura input
2. Ordinamento delle librerie (dipende dalla soluzione)
3. For per ogni libreria
    1. Modifica ordinamento (dipende dalla soluzione)
    2. Ricerca libri che danno lo score piu alto
    3. Salvataggio ID dei libri
4. Calcolo dello score in base a gli ID dei libri scansionati
5. Salvataggio output

COMPLESSITA

- Il tempo minimo per la risoluzione di tutti i casi di test varia da 2min a circa 9 min
- Un scambio del ordine delle biblioteche con tempo di registrazione diverso implica un calcolo del punteggio di tutte le biblioteche successive
- La soluzione classificata al primo posto ha raggiunto 27203691 punti

IMPLEMENTAZIONE

Soluzione 1 : Base

1. Lettura Input da file di testo
2. Registro le librerie sulla piattaforma con l ordine ricevuto
3. For per ogni libreria
    1. Cerco gli n libri che danno lo score piu alto
    2. Salvataggio ID dei libri
4. Calcolo lo score
5. Creo l output

Risultato pari a 10 mln

```
a_example
-  Punteggio:          21            (100 %)
                totale biblioteche iscritte:          2  s
b_read_on
-  Punteggio:          4126100       (87 %)
                totale biblioteche iscritte:          87
c_incunabula
-  Punteggio:          870640        (1 %)
                totale biblioteche iscritte:          198
d_tough_choices
-  Punteggio:          4109300       (50 %)
                totale biblioteche iscritte:          1500
e_so_many_books
-  Punteggio:          755788        (3 %)
                totale biblioteche iscritte:          36
f_libraries_of_the_world
-  Punteggio:          916843        (0 %)
                totale biblioteche iscritte:          5  s

totale punteggio: 10778692
```

Soluzione 2 : con euristica

1. Lettura Input da file di testo
2. Ordino le biblioteche per rapporto decrescente num_libri/Registrazione

```
sys.stdout.write("\r Ordinamento...")
Euristica = np.vectorize(lambda A, B: A/B)
punteggio_totale = Euristica(numero_libri, registrazione)

punteggio_totale, biblioteche_ordinate = ordinamento(punteggio_totale, biblioteche)
```

3. Scannerizzo per ogni libreria gli n libri con lo score maggiore
4. Calcolo il punteggio
5. Creo l output

```
a_example
- Punteggio:        21          (100 %)
              totale biblioteche iscritte:        2  su   2
b_read_on
- Punteggio:        5822900     (91 %)
              totale biblioteche iscritte:        91  su   100
c_incunabula
- Punteggio:        5586053     (13 %)
              totale biblioteche iscritte:        1310  su   10000
d_tough_choices
- Punteggio:        4812730     (50 %)
              totale biblioteche iscritte:        15001  su   30000
e_so_many_books
- Punteggio:        4601180     (13 %)
              totale biblioteche iscritte:        133  su   1000
f_libraries_of_the_world
- Punteggio:        5317660     (1 %)
              totale biblioteche iscritte:        18  su   1000

totale punteggio: 26140544
```

Soluzione 3 : ordinando punteggio totale su registrazione

1. Lettura Input da file di testo
2. Associo a ogni biblioteca uno punteggi definito come

$$Punteggio[biblioteca] = \frac{\sum_{i=0}^{k-1} Punteggio[i]}{Registrazione[biblioteca]}$$

Dove:

$$GiorniDisponibili[biblioteca] = D - Registrazione[biblioteca]$$
$$K = GiorniDisponibili[biblioteca] * MaxLibriScan[biblioteca]$$

3. Ordina librerie in base allo score
4. Per ogni biblioteca seleziono i libri con punteggio maggiore scannerizzabili
5. Creo l'output

Soluzione 3 : RISULTATO

- Un punteggio di circa 26,5 milioni

```
biblioteca_punteggio_libri = np.vectorize(lambda biblioteca: biblioteca.top_punteggio_libri())
biblioteche_scores = biblioteca_punteggio_libri(biblioteche)

Euristica = np.vectorize(lambda book_score, signup_days: book_score / signup_days)
punteggio_totale = Euristica(biblioteche_scores, registrazione)

punteggio_totale, biblioteche_ordinate = ordinamento(punteggio_totale, biblioteche)
```

```
a_example
-  Punteggio:          21      (100 %)
                totale biblioteche iscritte:        2   su
b_read_on
-  Punteggio:          5822900 (91 %)
                totale biblioteche iscritte:        91   su
c_incunabula
-  Punteggio:          5645747 (12 %)
                totale biblioteche iscritte:        1299   su
d_tough_choices
-  Punteggio:          4812730 (50 %)
                totale biblioteche iscritte:        15001   s
e_so_many_books
-  Punteggio:          5019670 (13 %)
                totale biblioteche iscritte:        138   su
f_libraries_of_the_world
-  Punteggio:          5240161 (1 %)
                totale biblioteche iscritte:        18   su

totale punteggio: 26541229
```

SWAP

```
def swap(lista, gruppi1 , unita1 , setoff1):

    gruppi=int(gruppi1)
    unita=int(unita1)

    setoff=int(setoff1)

    lista_modify=list(lista)

    for i in range(gruppi-1):
        max=i*setoff+unita
        if max<len(lista_modify):
            for j in range(unita):
                tmp = lista_modify[i*setoff]
                lista_modify[i*setoff] =lista_
                lista_modify[i*setoff+j]=tmp
    lista2=tuple(lista_modify)
    return lista2
```

Dopo aver ordinato le biblioteche decidiamo di modificare l'ordinamento tramite la seguente funzione dove :

- **Lista** = array base
- **Gruppi** = numero di gruppi
- **Unita** = unita che scambio per ogni gruppo
- **Setoff**= scambio l elementi i-esimo con quello nella posizione i+setoff

Soluzione 1 : MIGLIORAMENTO punteggio con swap

- Divido le biblioteche in Gruppi
- Per ogni gruppo ha 25 biblioteche
- Seleziono le prime 19 biblioteche per ogni gruppo
- E le scambio con quelle che si trovano 42 posizioni successive

Abbiamo un miglioramento di 76583 punti

```
Gruppi=len(biblioteche)/25
unita_gruppo=len(biblioteche)/Gruppi/1.3
setoff=unita_gruppo*1.7
biblioteche1=swap(biblioteche,Gruppi,unita_gruppo,setoff)
```


```
a_example
-  Punteggio:        21          (100 %)
                totale biblioteche iscritte:        2  su   2
b_read_on
-  Punteggio:        4141900     (87 %)
                totale biblioteche iscritte:        87  su   100
c_incunabula
-  Punteggio:        866531      (1 %)
                totale biblioteche iscritte:        197  su   10000
d_tough_choices
-  Punteggio:        4109300     (50 %)
                totale biblioteche iscritte:        15001  su   30000
e_so_many_books
-  Punteggio:        746992      (3 %)
                totale biblioteche iscritte:        36  su   1000
f_libraries_of_the_world
-  Punteggio:        990531      (0 %)
                totale biblioteche iscritte:        5  su   1000

totale punteggio: 10855275
```

Soluzione 1 : PEGGIORAMENTO punteggio con scambio

- Divido le biblioteche in Gruppi
- Ogni gruppo ha 20 biblioteche
- Per ogni gruppo scambio le biblioteche con quelle che si trovano 25 posizioni successive

Ottenendo un peggioramento di 72141 punti

```
Gruppi=len(biblioteche)/20
unita_gruppo=len(biblioteche)/Gruppi
setoff=unita_gruppo*1.2
biblioteche1=swap(biblioteche,Gruppi,unita_gruppo,setoff)
```

```
a_example
-  Punteggio:        21              (100 %)
                totale biblioteche iscritte:        2  su   2
b_read_on
-  Punteggio:        4124900         (87 %)
                totale biblioteche iscritte:        87  su   100
c_incunabula
-  Punteggio:        867256          (1 %)
                totale biblioteche iscritte:        197  su   10000
d_tough_choices
-  Punteggio:        4109300         (50 %)
                totale biblioteche iscritte:        15001  su   30000
e_so_many_books
-  Punteggio:        771682          (3 %)
                totale biblioteche iscritte:        36  su   1000
f_libraries_of_the_world
-  Punteggio:        833391          (0 %)
                totale biblioteche iscritte:        4  su   1000

totale punteggio: 10706550
```

Soluzione 2 : PEGGIORAMENTO con scambio

- Divido le biblioteche in Gruppi
- Ogni gruppo ha 30 biblioteche
- Selezione le prime 15 biblioteche per ogni gruppo
- E le scambio con quelle che si trovano 24 posizioni successive

Abbiamo un peggioramento di 100 mila punti

```
punteggio_totale, biblioteche_ordinate = ordinamento(punteggio_totale, biblioteche)
Gruppi=len(biblioteche)/30
unita_gruppo=len(biblioteche)/Gruppi/2
setoff=unita_gruppo*1.6

biblioteche_ordinate_2=swap(biblioteche_ordinate,Gruppi,unita_gruppo,setoff)
```

```
a_example
-  Punteggio:          21          (100 %)
                totale biblioteche iscritte:        2  su   2
b_read_on
-  Punteggio:          5815700     (91 %)
                totale biblioteche iscritte:        91  su   100
c_incunabula
-  Punteggio:          5586053     (13 %)
                totale biblioteche iscritte:        1310  su   10000
d_tough_choices
-  Punteggio:          4813445     (50 %)
                totale biblioteche iscritte:        15001  su   30000
e_so_many_books
-  Punteggio:          4600802     (13 %)
                totale biblioteche iscritte:        133  su   1000
f_libraries_of_the_world
-  Punteggio:          5224557     (1 %)
                totale biblioteche iscritte:        17  su   1000

totale punteggio: 26040578
```

Soluzione 2 : MIGLIORAMENTO con scambio

È possibile notare un miglioramento di 40 punti

- Divido le librerie in Gruppi
- Per ogni gruppo seleziono 15 biblioteche
- E le scambio con quelle che si trovano 22 posizioni successive

Notiamo un miglioramento di circa 40 punti

```
punteggio_totale, biblioteche_ordinate = ordinamento(punteggio_totale, biblioteche
Gruppi=len(biblioteche)/40
unita_gruppo=len(biblioteche)/Gruppi/10
setoff=len(biblioteche)*1.5
```

```
a_example
-  Punteggio:        21          (100 %)
              totale biblioteche iscritte:        2  su   2
b_read_on
-  Punteggio:        5822700     (91 %)
              totale biblioteche iscritte:        91  su   100
c_incunabula
-  Punteggio:        5586053     (13 %)
              totale biblioteche iscritte:        1310  su   10000
d_tough_choices
-  Punteggio:        4812730     (50 %)
              totale biblioteche iscritte:        15001  su   30000
e_so_many_books
-  Punteggio:        4601420     (13 %)
              totale biblioteche iscritte:        133  su   1000
f_libraries_of_the_world
-  Punteggio:        5317660     (1 %)
              totale biblioteche iscritte:        18  su   1000

totale punteggio: 26140584
```

ORDIAMENTO ITERATIVO

```
a=5 # distanza gruppi ordinamento
b=4 # elementi ordinati
SALVATAGGIO FILE
cnt=10 # spiazzamento inizio
sx=dx=1000
temp = 0
libri_totali = set()
with open("outputs/" + file + ".out", 'w+') as f:
    f.write(str(tot_biblio) + "\n")
    giorno = 0
    for i in range(tot_biblio) :
        if giorno<=D:

            if i==cnt:
                cnt+=a
                sx=cnt-a+1
                dx=cnt-a+b+1
                Euristica = np.vectorize(lambda biblioteca: biblioteca.punteggio_iterativo())
                punteggio_totale = Euristica(biblioteche_ordinate[i+1:i+b+1])
                punteggio_totale_2, biblioteche_ordinate_2 = ordinamento(punteggio_totale, biblioteche_ordinate[i+1:i+b+1])

            if i>=sx and i<dx :
                biblioteca_attuale =  biblioteche_ordinate_2[i-sx]
            else :
                biblioteca_attuale = biblioteche_ordinate[i]

            chosen_id_libri = biblioteca_attuale.top_libri_id(giorno)
            libri_totali.update(chosen_id_libri)
            giorno += biblioteca_attuale.registrazione
```

Dopo aver selezionato le prime **(cnt)** librerie ogni **(a)** libreria ordino **(b)** librerie secondo un euristica che tiene conto anche dei libri inseriti precedentemente cosi da selezionare la libreria con lo score maggiore

Soluzione 3 : Miglioramento con ordinamento iterativo

```python
def punteggio_iterativo(self, start_day=0):
    global punteggio_libri, registrazione, biblioteche_max_books_scanned_su_day, D, libri_totali

    giorni_disponibili = D - self.registrazione - giorno
    libri_disponibili = list(set(self.id_libri) - set(libri_totali))
    # NUMERO MASSIMO DI LIBRI SCANSIONABILI
    max_num_libri_scan = max(min(int(giorni_disponibili * self.scanned_su_day), len(self.id_libri)), 0)

    punteggio_attuale = np.take(punteggio_libri, self.id_libri)

    # get top k books (k=max_num_books)
    ind = np.argpartition(punteggio_attuale, -max_num_libri_scan)[-max_num_libri_scan:]
    best_books_scores = np.take(punteggio_attuale, ind)
    totale=np.sum(best_books_scores)
    return totale
```

Sottraiamo ai libri della biblioteca i libri gia scansionati
Calcoliamo i giorni rimanente in base al giorno di quando abbiamo chiamato la funzione
Con questi due valori aggiornati calcoliamo il nuovo punteggio delle librerie

```
a_example
-  Punteggio:         21      (100 %)
                totale biblioteche iscritte:         2  su   2
b_read_on
-  Punteggio:         5821100 (91 %)
                totale biblioteche iscritte:         91  su   100
c_incunabula
-  Punteggio:         5646000 (12 %)
                totale biblioteche iscritte:         1299  su   10000
d_tough_choices
-  Punteggio:         4812665 (50 %)
                totale biblioteche iscritte:         15001  su   30000
e_so_many_books
-  Punteggio:         5021498 (14 %)
                totale biblioteche iscritte:         140  su   1000
f_libraries_of_the_world
-  Punteggio:         5253737 (1 %)
                totale biblioteche iscritte:         18  su   1000

totale punteggio: 26555021
```

- Miglioramento di 13792 punti

RECAP SOLUZIONI

Soluzione	Soluzione	Punteggio
0	Top 1 solution hash code	26.555.021
1	Senza ordinamento	10.778.692
1.1	Senza ordinamento + scambio_down	10.706.550
1.2	Senza ordinamento + scampio_up	10.855.275
2	Num_Libri/registrazione	26.140.544
2.1	Num_Libri/registrazione + scambio down	26.040.568
2.2	Num_Libri/registrazione + scambio up	26.140.584
3	Sum_punteggio_libri/registrazione	26.541.229
3.1	Sum_punteggio_libri/registrazione + ordinamento iterativo	26.555.021

CODICE

CODICE (1/1)

LETTURA DA FILE :

```
for file in files:

## LETTURA DA FILE
    with open("inputs/" + file + ".txt", "r") as f:
        content = f.read().splitlines()
    print(file)

    libri_esistenti, tot_biblio, D = list(map(int, content[0].split(' ')))

    punteggio_libri = list(map(int, content[1].split(' ')))
    pos = 1
    numero_libri = np.zeros(tot_biblio)
    registrazione = np.zeros(tot_biblio)
    biblioteche_scanned_su_day = np.zeros(tot_biblio)
    biblioteche = np.empty(tot_biblio, dtype=Biblioteca)
    for i in range(tot_biblio):
        pos += 1
        n, t, m = list(map(int, content[pos].split(' ')))
        numero_libri[i] = n
        registrazione[i] = t
        biblioteche_scanned_su_day[i] = m

        pos += 1
        id_libri = np.asarray(list(map(int, content[pos].split(' '))))
        biblioteche[i] = Biblioteca(i, id_libri, t, m)
```

SALVATAGGIO SOLUZIONE :

```
### SALVATAGGIO FILE
    libri_totali = set()
    with open("outputs/" + file + ".out", 'w+') as f:

        f.write(str(tot_biblio) + "\n")

        giorno = 0
        for i in range(tot_biblio):
            if giorno<=D:
                biblioteca_attuale = biblioteche[i]
                chosen_id_libri = biblioteca_attuale.top_libri_id(giorno)
                libri_totali.update(chosen_id_libri)
                giorno += biblioteca_attuale.registrazione

                if len(chosen_id_libri) > 0:
                    f.write(str(biblioteca_attuale.id) + " " + str(len(chosen_id_libri)) + "\n")
                    f.write(str(' '.join(map(str, chosen_id_libri))) + "\n")
                else:
                    f.write(str(biblioteca_attuale.id) + " 1\n")
                    f.write(str(biblioteca_attuale.id_libri[0])+"\n")
                c=i+1
                progress = 100 * (i+1) / (tot_biblio)
                sys.stdout.write("\r Caricamento                    (" + str(int(progress)) + " %)")

        punteggio = somma_punteggio_libri(libri_totali)
        totale_punteggio += punteggio

        print("\r-  Punteggio:        ", punteggio)
        print("                totale biblioteche iscritte:        ",  c  , " su  " , tot_biblio)

print("")
print("totale punteggio:", totale_punteggio)
```

CODICE (2/2)

ORDINAMENTO SOLUZIONE 2:

```
## INIZIO PROGRAMMA
    sys.stdout.write("\r Ordinamento...")
    Euristica = np.vectorize(lambda A, B: A/B)
    punteggio_totale = Euristica(numero_libri, registrazione)

    punteggio_totale, biblioteche_ordinate = ordinamento(punteggio_totale, biblioteche)
```

ORDINAMENTO SOLUZIONE 3:

```
## INIZIO PROGRAMMA
    sys.stdout.write("\rOrdinamento...")

    biblioteca_punteggio_libri = np.vectorize(lambda biblioteca: biblioteca.top_punteggio_libri())
    biblioteche_scores = biblioteca_punteggio_libri(biblioteche)

    Euristica = np.vectorize(lambda book_score, signup_days: book_score / signup_days)
    punteggio_totale = Euristica(biblioteche_scores, registrazione)

    punteggio_totale, biblioteche_ordinate = ordinamento(punteggio_totale, biblioteche)
```

FUNZIONE LIBRI CON MAGGIOR PUNTEGGIO:

```
#definisco il miglior libro
    def top_libri_id(self, giorno=0):
        global punteggio_libri, registrazione, biblioteche_max_books_scanned_su_day, D, libri_totali

        giorni_disponibili = D - self.registrazione - giorno
        libri_disponibili = list(set(self.id_libri) - set(libri_totali))
        punteggio_attuale = np.take(punteggio_libri, libri_disponibili)

        max_num_libri_scan = max(min(int(giorni_disponibili * self.scanned_su_day), len(libri_disponibili)), 0)
        if max_num_libri_scan == 0:
            return []
        ind = np.argpartition(punteggio_attuale, -max_num_libri_scan)[-max_num_libri_scan:]

        return np.take(libri_disponibili, ind)
```

FUNZIONE LIBRI CON MIGLIOR PUNTEGGIO:

```
def top_punteggio_libri(self, giorno=0):
    global punteggio_libri, registrazione, biblioteche_max_books_scanned_su_day, D
    giorni_disponibili = D - self.registrazione - giorno
    max_num_libri_scan = max(min(int(giorni_disponibili * self.scanned_su_day), len(self.id_libri)), 0)

    punteggio_attuale = np.take(punteggio_libri, self.id_libri)
    ind = np.argpartition(punteggio_attuale, -max_num_libri_scan)[-max_num_libri_scan:]

    best_books_scores = np.take(punteggio_attuale, ind)

    return np.sum(best_books_scores)
```

ULTERIORI IMPLEMENTAZIONI

SIMULATED ANNEALING

Partiamo da una soluzione corrente *x* poi genera una nuova soluzione x' tramite una mossa. x' se è migliore allora la sostituisco con x, altrimenti decido se accettare o meno attraverso una funzione di probabilistica, con l aumentare delle iterazioni ha meno probabilità di accettare la soluzione peggiore

In un euristica si simulated annealing si aggiorna t_k in maniera probabilistica dove :

$$P(accettazione\ di\ x') = \begin{cases} 1 & se\ f(x') \leq f(x) \\ e^{\frac{-f(x')-f(x)}{t_k}} & se\ f(x') > f(x) \end{cases}$$

t_k viene abbassato nel corso dell'algoritmo abbassando la probabilità di accettazione delle soluzioni peggiorative
t_k viene abbassato 'lentamente' la probabilità di raggiungere l'ottimo è pari a 1

SCELTA T_0

Si determina una temperatura T_0 in modo tale che tutte le transazioni siano accettate e quindi $e^{-\frac{\Delta f}{T}} \approx 1\ \ \forall \Delta f > 0$

- si considera il valore di *funzione obiettivo* f_0 della soluzione iniziale
- si pone un valore di riferimento per la *massima variazione della funzione obiettivo* (ad esempio $\Delta f_{max} = |f_0/2|$)
- si pone $T_0 = 10 \cdot \Delta f_{max} = 10 \cdot |f_0/2|$

In questo modo ci si assicura che, per transizioni caratterizzate da un *incremento di funzione obiettivo pari al 50%* del valore iniziale, la *probabilità di accettazione è praticamente unitaria.*

SIMULTED ANNEALING : T_f e decremento di T

SCELTA T_f

Il valore del parametro T_f deve essere scelto in modo tale che, la probabilità di *accettazione di soluzioni peggiorative* deve essere praticamente *nulla*. Deve dunque risultare:

$$e^{-\frac{\Delta f}{T}} \approx 0 \quad \forall \Delta f > 0 \quad ossia \ \frac{\Delta f}{T} \to \infty \quad \forall \Delta f > 0$$

Per garantire questa condizione si può ad esempio operare nel seguente modo:

- si considera il valore di *funzione obiettivo* f_0 della soluzione iniziale
- si pone un valore di riferimento per la *minima variazione della funzione obiettivo* (ad esempio $\Delta f_{min} = 10^{-3}|f_0|$)
- si pone $T_f = 10^{-1} \cdot \Delta f_{min} = 10^{-4} \cdot |f_0|$

In questo modo ci si assicura che, per transizioni caratterizzate da un *incremento di funzione obiettivo pari allo 0,1%* del valore iniziale, la *probabilità di accettazione è praticamente nulla.*

SCELTA DECREMTO DI T

La regola più frequentemente utilizzata per decrementare il parametro T dalla generica iterazione k alla successiva $k+1$ è $T_{k+1} = \alpha\ T_k$ con $\alpha < 1$

I valori di L_k vanno *scelti di conseguenza*:

- Se α è *elevato* (ad esempio *0.99*) il decremento è più lento e quindi L_k può essere basso
- Se α è *basso* (ad esempio *0.8*) L_k deve essere *più elevato* per *ripristinare le condizioni di equilibrio termico*
- Una possibile soluzione è quella di considerare $L_k = L$ costante e correlato alle *dimensioni del problema* n (ad esempio $L = n$).

In altre proposte L_k viene scelto in modo da assicurare che *per ogni valore di* T_k il *numero delle transizioni accettate sia almeno pari ad un certo valore* μ_{min}

ALGORITMO

ALGORITMO

1) **INIZIALIZZAZIONE** : trovare la soluzione S iniziale

   Ordino le librerie determino il punteggio e salvo l'ordine delle biblioteche registrate

2) **DEFINIZIONE MOSSA** : scelta casual di una soluzione S' appartenente ad S

   Scelgo un insieme S' dove non è presente nessuna libreria selezionata precedentemente

   Definisco T_0 tale che all'inizio la probabilità di accettazione è pari ad 1

   Definisco T_f tale che all'inizio la probabilità di accettazione è pari ad 0

   Definisco L uguale al numero di biblioteche registrate nella soluzione precedente quindi un

3) **ACCETTAZIONE DELLA MOSSA:**

$$P(accettazione\ di\ x') = \begin{cases} 1 & se \Delta \leq 0 \\ e^{\frac{-\Delta f}{t_k}} & se\ \Delta f > 0 \end{cases}$$

se ***Δf*** *è minore o uguale di 0,* ***S'*** *viene accettata come soluzione corrente*

se Δf *è maggiore di 0 si genera un numero causale tra 0 e 1; se questo è inferiore a* $e^{-\frac{\Delta f}{T}}$, ***S'*** *viene accettata altrimenti la soluzione corrente resta inalterata*

POSSIBILE ALGORITMO DI SIMULATE ANNEALING

```
#DOPO AVER CALCOLATO LA PRIMA SOLUZIONI
S1=Biblioteche_ordinate[num_biblioteche+1 : ]
S2= Biblioteche_ordinate[ : num_biblioteche ]
a=b=0
For i in range(tot_biblio):
        if giorno<=D:
        #
                punteggio_S1_attuale=S1[a]punteggio_iterativo()
                punteggio_S2_attuale= S2[b]punteggio_iterativo()
                deltaF=punteggio_S2_attuale-punteggio_S1_attuale

                if b<=len(S2):
                #CASO soluzione migliore
                        if deltaF>=0:
                                New_biblioteche_ordinate[i]=S[b]
                                biblioteca_attuale=S[b]
                                b+=1
                #caso soluzione peggiore se il valore uscito è minore della probabilita data dalla temperatura T allora scelgo la soluzione peggiore
                        Else :
                                probabilita=math.exp(-deltaF/T)
                                if random.random()<probabilita
                                        New_biblioteche_ordinate[i]=S[b]
                                        biblioteca_attuale=S[b]
                                        b+=1
                                else:
                                        New_biblioteche_ordinate[i]=S[a]
                                        a+=1
                        Decremento(T , T0 , Tf)
                else:
                                New_biblioteche_ordinate[i]=S[a]
                                a+=1
                ### CALCOLO PUNTEGGIO ###
```