ASD Laboratorio 02

Cristian Consonni/Lorenzo Ghiro

UniTN

31/10/2018

CALENDARIO

09/10	Introduzione
30/10	Ad-hoc
20/11	Grafi 1
27/11	Grafi 2
04/12	Progetto 1
11/12	Progetto 1

Progetto:

- 4 – 11 dicembre;
- Iscrizione dei gruppi ai progetti entro il **2 dicembre**:

  `http://bit.ly/ASDprog_2018-2019`

  (dovete essere loggati con l'account UniTN)

CALCOLO COMBINATORIO (I)

Quante sono le COPPIE senza ripetizioni e senza tenere conto dell'ordine da un insieme di n elementi?

NUMERO DI COMBINAZIONI DI 2 ELEMENTI DATI n SENZA RIPETIZIONI

$$C(n,2) = \binom{n}{2} = \frac{n \cdot (n-1)}{2}$$

Caso particolare del COEFFICIENTE BINOMIALE:

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Intuizione (numero di disposizioni di k elementi da n, diviso per il numero di permutazioni di k elementi):

$$C(n,k) = \frac{D(n,k)}{P(k)} = \frac{n!}{(n-k)!} \cdot \frac{1}{k!}$$

Considerando le ripetizioni:

NUMERO DI COMBINAZIONI DI 2 ELEMENTI DATI n CON RIPETIZIONE

$$C'(n,2) = \binom{n+1}{2} = \frac{n \cdot (n+1)}{2}$$

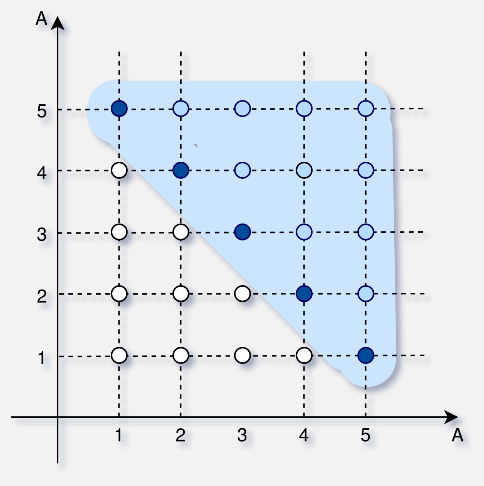
In questo caso:

$$C'(n,k) = \binom{n+k-1}{k} = \frac{(n+k-1)!}{(n-1)!k!}$$

Intuizione:

- corrispondenza biunivoca tra combinazione con ripetizioni e combinazioni senza ripetizione k elementi da $n+k-1$

CALCOLO COMBINATORIO (III)



$$C = \{(i,j) \mid i,j \in A \subseteq N, j \geq i\}$$

Soluzione $\mathcal{O}(N^2)$

Costruiamo array delle somme:

$$S_i = \sum_{j=1}^{i} A_j$$

Per ogni sottosequenza, calcoliamo la somma in $O(1)$:

$$\text{Somma da i a j} = S_j - S_{i-1}$$

$$S_i = \sum_{j=1}^{i} A_j$$

Esempio:

```
5
2
-3
4
1
5
```

Array delle somme:

`index`	1	2	3	4	5
`array`	2	-3	4	1	5
`S`	2	-1	3	4	9

Combinazioni (i, j):

i/j	1	2	3	4	5
1	2	-1	3	4	9
2	–	-3	1	2	7
3	–	–	4	5	**10**
4	–	–	–	1	6
5	–	–	–	–	5

La sottosequenza di somma massima conterrà un elemento con indice massimo, sia esso i:

- B_i la sottosequenza di somma massima che ha come ultimo elemento il numero in posizione i;
- assumendo di conoscere B_{i-1}, procedendo per induzione allora: $B_i = \max(A_i, B_{i-1} + A_i)$;
- terminiamo restituendo il valore massimo individuato durante l'induzione: $max(B_0, B_1, ..., B_{N-1})$.

SOTTOSEQ: SOLUZIONE $\mathcal{O}(N)$ (I)

ALGORITMO DI KADANE, $\mathcal{O}(N)$

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

Esempio:

```
5
2
-3
4
1
5
```

`index`	1	2	3	4	5
`array`	2	-3	4	1	5
`last`	2	-1	4	5	10
`mx`	2	2	4	5	**10**

Soluzione “a forza bruta” $\mathcal{O}\left((RC)^3\right) \sim \mathcal{O}(N^6)$:

1. Ci sono $(RC)^2 \sim N^4$ sottomatrici[1]
2. È possibile calcolare la somma di una sottomatrice in meno di $\mathcal{O}(RC) \sim \mathcal{O}(N^2)$?
3. Dobbiamo veramente guardare tutte le sottomatrici?

[1] (link di approfondimento)

CALCOLARE LA SOMMA IN $O(1)$

Stessa idea di prima. Riempiamo un array somma ($O(RC)$)

$$S[i,j] = \sum_{a=1}^{i} \sum_{b=1}^{j} A[a,b]$$

Per calcolare la somma da $[r_1, c_1]$ a $[r_2, c_2]$:

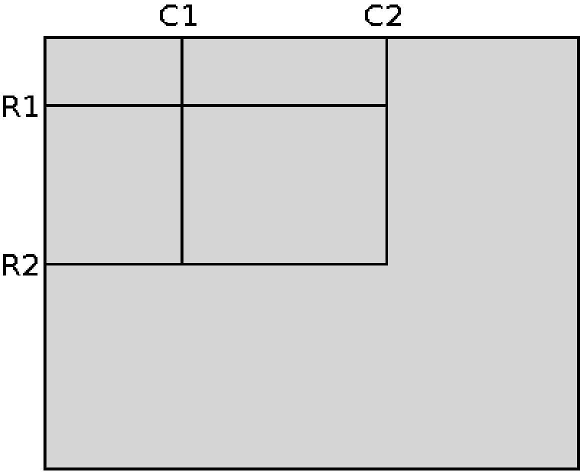
$$S[r_2, c_2] + S[r_1, c_1] - S[r_2, c_1] - S[r_1, c_2]$$

Sfruttando questa idea otteniamo un algoritmo $O((RC)^2)$.

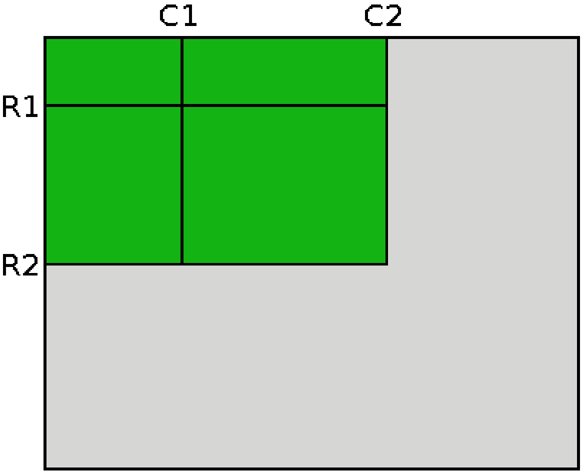
NOTA IMPLEMENTATIVA

Creando $S[i,j]$ con un "orlo" di zeri si semplifica la gestione degli indici.

SOTTOMAT: MATRICE DELLE SOMME (II)

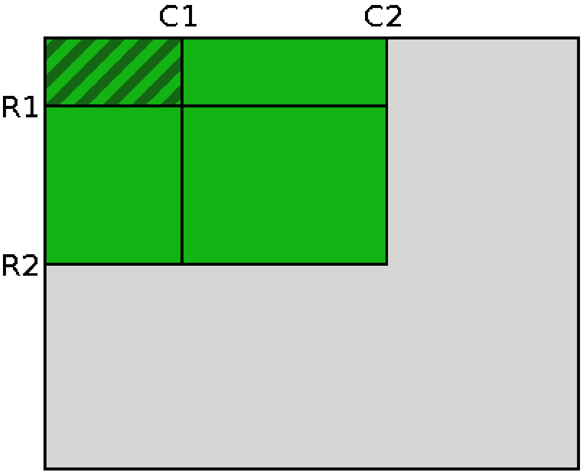


SOTTOMAT: MATRICE DELLE SOMME (III)

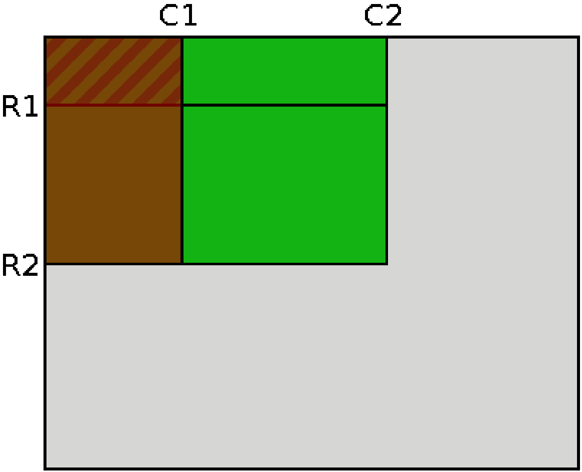


$S[r_2, c_2]$

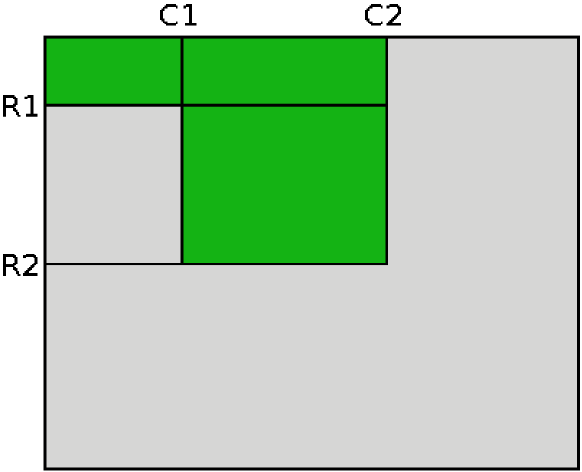
SOTTOMAT: MATRICE DELLE SOMME (IV)



$$S[r_2, c_2] + S[r_1, c_1]$$

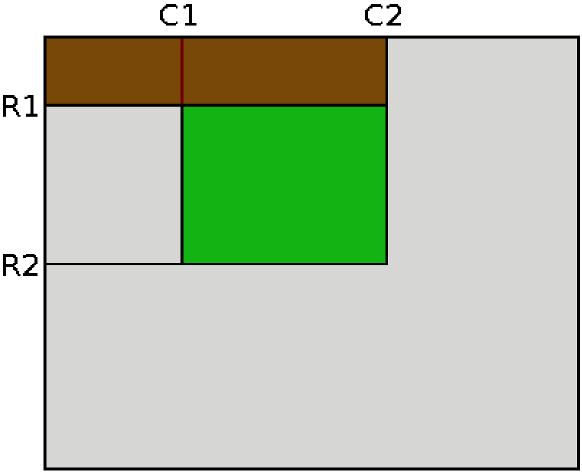


$$S[r_2, c_2] + S[r_1, c_1] - S[r_2, c_1]$$

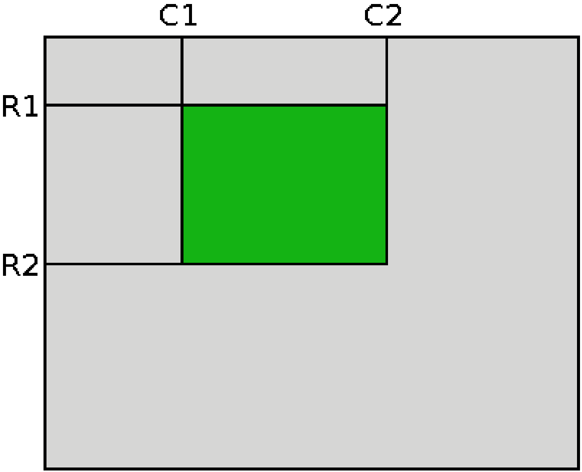


$$S[r_2, c_2] + S[r_1, c_1] - S[r_2, c_1]$$

SOTTOMAT: MATRICE DELLE SOMME (VII)



$$S[r_2, c_2] + S[r_1, c_1] - S[r_2, c_1] - S[r_1, c_2]$$



$$S[r_2, c_2] + S[r_1, c_1] - S[r_2, c_1] - S[r_1, c_2]$$

SOLUZIONE $\mathcal{O}\left((RC)^2\right) \sim \mathcal{O}(N^4)$

- per ogni coppia di righe $r_s, r_e \rightarrow \mathcal{O}\left(R^2\right)$
- per ogni coppia di colonne $c_s, c_e \rightarrow \mathcal{O}\left(C^2\right)$

$\Rightarrow$ calcoliamo la somma $\rightarrow \mathcal{O}(1)$:

$$S[r_s, c_s] + S[r_e, c_e] - S[r_e, c_s] - S[r_s, c_e]$$

- possiamo sfruttare la soluzione ottima $O(N)$ del problema della sottosequenza di somma massima per trovare la sottomatrice di somma massima?
- consideriamo tutte le sottomatrici che partono dalla colonna$^{(*)}$ C_1 e arrivano alla colonna C_2, possiamo applicare la sottosequenza di somma massima?
  - se $C_1 = C_2$, stiamo considerando una singola colonna, possiamo applicare facilmente la sottosequenza di somma massima
  - negli altri casi?

$(*)$ il discorso è speculare per righe e colonne

Per ogni coppia C_1, C_2 creiamo un'istanza del problema della sottosequenza di somma massima.

$\Rightarrow$ se $C_1 = C_2$:

$C_1 = C_2 = 2$

/	1	2	3	4
1	2	-9	2	3
2	1	4	5	1
3	-2	3	4	1

$\rightsquigarrow$

	$(r_1, c_1), (r_2, c_2)$		
-9	$(1,2),(1,2)$		
4	$(1,2),(2,2)$	$(2,2),(2,2)$	
3	$(1,2),(3,2)$	$(2,2),(3,2)$	$(3,2),(3,2)$

Con Kadane riusciamo a considerare tutte e 6 le possibili sottomatrici.

Per ogni coppia C_1, C_2 creiamo un'istanza del problema della sottosequenza di somma massima.

$\Rightarrow$ se $C_1 \neq C_2$:

$C_1 = 2; C_2 = 4$

/	1	2	3	4
1	2	-9	2	3
2	1	4	5	1
3	-2	3	4	1

$\rightsquigarrow$

	$(r_1, c_1), (r_2, c_2)$		
-4	$(1,2),(1,4)$		
10	$(1,2),(2,4)$	$(2,2),(2,4)$	
8	$(1,2),(3,4)$	$(2,2),(3,4)$	$(3,2),(3,4)$

Con Kadane troviamo che la sottosequenza massima è 18 $(10 + 8)$ e corrisponde alla sottomatrice $(2,2),(3,4)$

Soluzione $\mathcal{O}(N^3)$

Per ogni coppia di colonne C_1,C_2:

1. Costruiamo l'array $S[1..R]$, di dimensione pari al numero di righe R;
2. Inseriamo in $S[i]$ *"la somma degli elementi appartenenti alla riga i e compresi fra le colonne C_1,C_2"*. In formula: $S[i] = \sum_{j=C_1}^{C_2} A[i][j]$;
3. Usiamo l'algoritmo lineare per la sottosequenza di somma massima su S.

$\Rightarrow$ $\mathcal{O}(RC^2)$, oppure $\mathcal{O}(R^2C)$

I sorgenti sono disponibili su
`http://judge.science.unitn.it/slides/`

Problemi

Testi completi su `https://judge.science.unitn.it/`.

Sorting

Implementate un algoritmo di ordinamento $\mathcal{O}(N \log N)$.

Intervalli

Dato un insieme di intervalli temporali, scoprire il periodo più lungo non coperto da alcun intervallo.

Sorting pesato

Avete un array di N interi, con i numeri da 1 a N (in ordine sparso). Ad ogni turno potete scambiare le posizioni di due interi, pagando la loro somma. Qual è il numero minimo di turni per ordinare l'array? Quant'è il prezzo minimo?

Potete definire la matrice somma $S[i,j]$ nel modo seguente:

```
for(int i=0;i<R;i++){
  for(int j=0;j<C;j++){
    in>>A[i][j];
    if(i==0){
      if(j==0) {
        S[i][j]=A[i][j];
    ...
        S[i][j]=S[i][j-1] + \
                S[i-1][j] - \
                S[i-1][j-1] + \
                A[i][j];
    ...
```

ma esiste un modo più furbo che vi semplifica la vita.