ASD Laboratorio 03

Cristian Consonni/Lorenzo Ghiro

UniTN

20/11/2017

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`

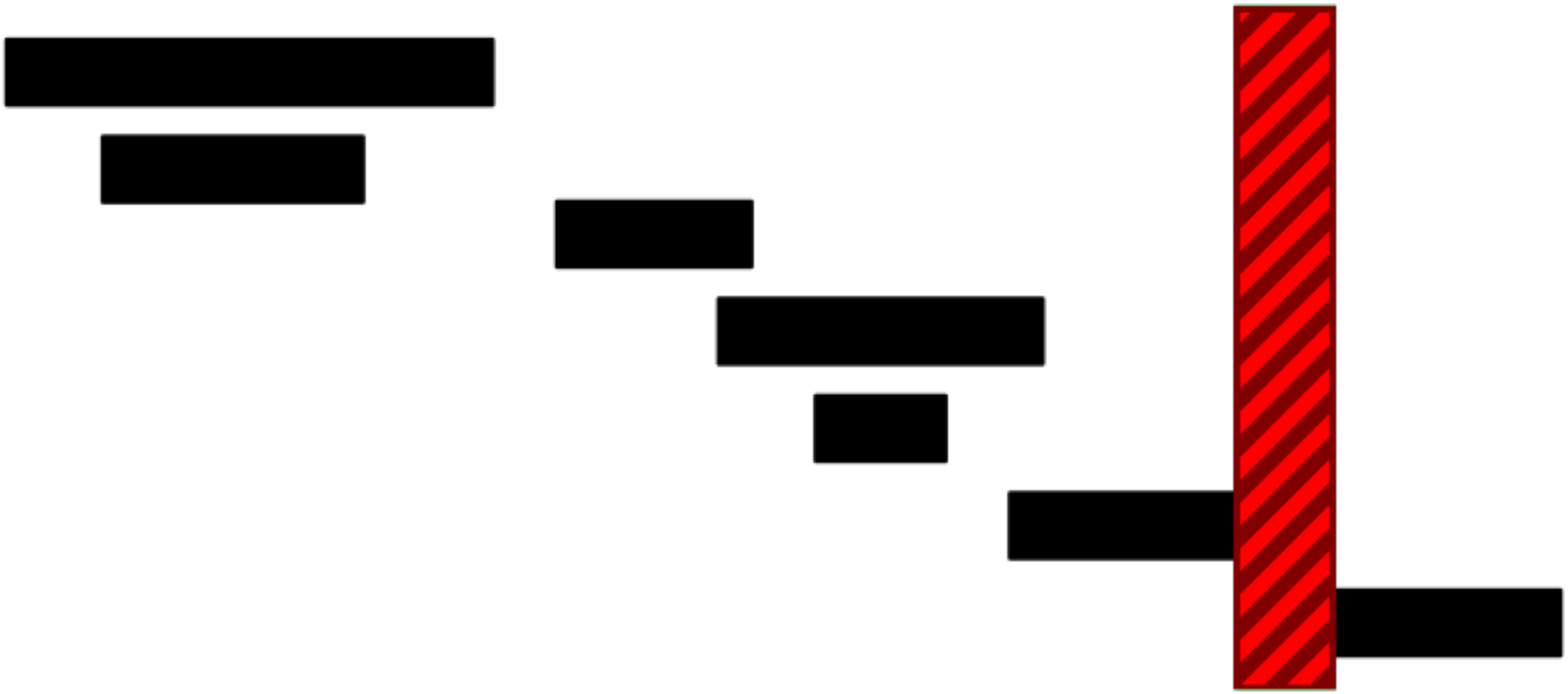
  (dovete essere loggati con l'account UniTN)

Merge sort

```
vector<int> merge_sort(vector<int>& arr){
  if(arr.size()<=1)
    return arr;

  // Spezzo l'array in due parti
  int m=arr.size()/2;

  vector<int> left(&arr[0],&arr[m]);
  vector<int> right(&arr[m],&arr[arr.size()]);

  // Ordino le due parti
  left=merge_sort(left);
  right=merge_sort(right);
```

```
  // Inserisco guardie
  left.push_back(INT_MAX);
  right.push_back(INT_MAX);

  vector<int> result;
  int l=0,r=0;
  for(int i=0; i<arr.size(); i++){
    if(right[r]<left[l]){
      result.push_back(right[r]);
      r++;
    } else {
      result.push_back(left[l]);
      l++;
    }
  }
  return result;
}
```

Dati N intervalli $\{(Inizio_i, Fine_i)\}_{i=1}^{N}$, vogliamo sapere quale è il piú lungo periodo non coperto da alcun intervallo, considerando solo gli istanti compresi fra il minimo istante di inizio ed il massimo istante di fine degli intervalli.



SOLUZIONI: INTERVALLO (I)

SIMULAZIONE $\mathcal{O}(M)$

1. Se gli intervalli arrivano fino all'istante M, creare un array $A[1..M]$ di booleani.
2. Per ogni intervallo, segnare gli istanti coperti dall'intervallo.
3. Visitare l'array per trovare i periodi scoperti.

Esempio:

```
4
2 5
8 12
1 4
7 9
```

1	2	3	4	5	6	7	8	9	10	11	12

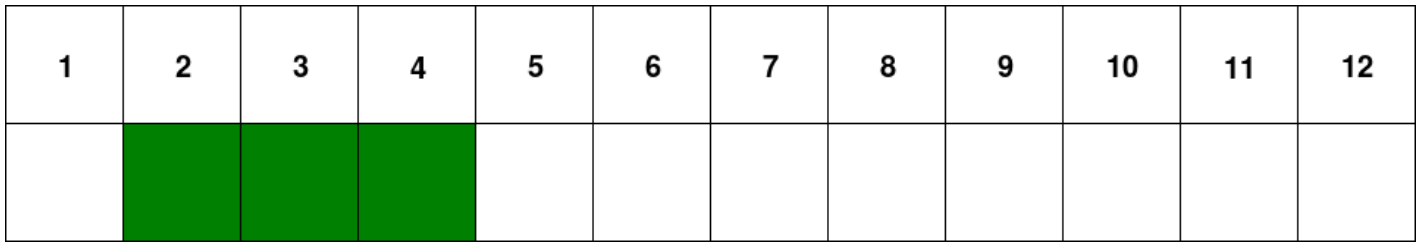
Soluzioni: intervallo (I)

Simulazione $\mathcal{O}(M)$

1. Se gli intervalli arrivano fino all'istante M, creare un array $A[1..M]$ di booleani.
2. Per ogni intervallo, segnare gli istanti coperti dall'intervallo.
3. Visitare l'array per trovare i periodi scoperti.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



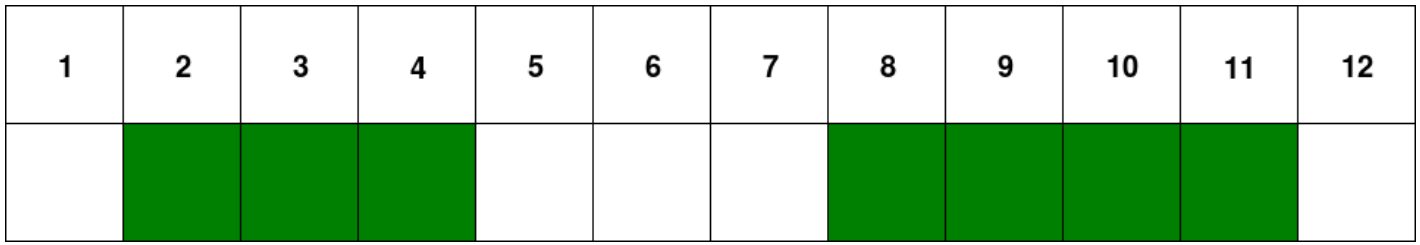
Soluzioni: intervallo (I)

Simulazione $\mathcal{O}(M)$

1. Se gli intervalli arrivano fino all'istante M, creare un array $A[1..M]$ di booleani.
2. Per ogni intervallo, segnare gli istanti coperti dall'intervallo.
3. Visitare l'array per trovare i periodi scoperti.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



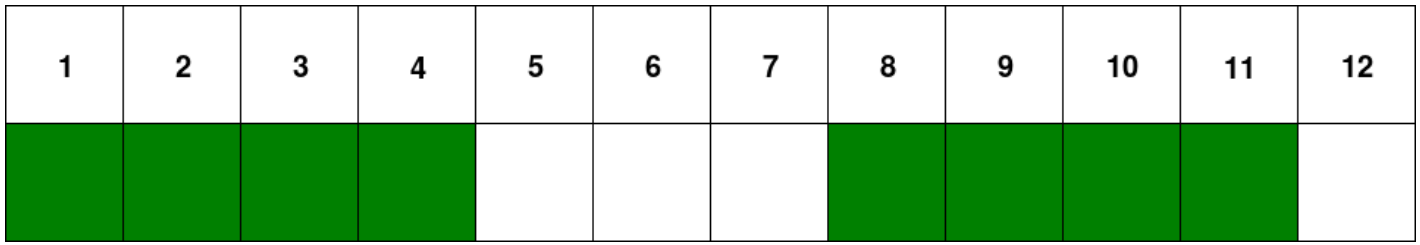
Soluzioni: intervallo (I)

Simulazione $\mathcal{O}(M)$

1. Se gli intervalli arrivano fino all'istante M, creare un array $A[1..M]$ di booleani.
2. Per ogni intervallo, segnare gli istanti coperti dall'intervallo.
3. Visitare l'array per trovare i periodi scoperti.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



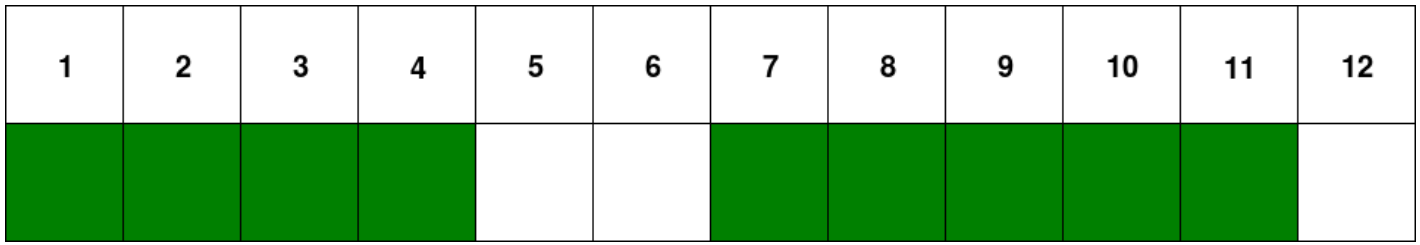
SOLUZIONI: INTERVALLO (I)

SIMULAZIONE $\mathcal{O}(M)$

1. Se gli intervalli arrivano fino all'istante M, creare un array $A[1..M]$ di booleani.
2. Per ogni intervallo, segnare gli istanti coperti dall'intervallo.
3. Visitare l'array per trovare i periodi scoperti.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



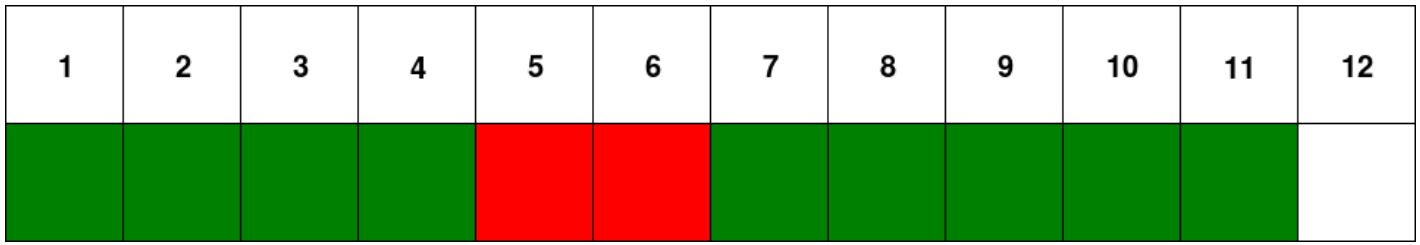
Soluzioni: intervallo (I)

Simulazione $\mathcal{O}(M)$

1. Se gli intervalli arrivano fino all'istante M, creare un array $A[1..M]$ di booleani.
2. Per ogni intervallo, segnare gli istanti coperti dall'intervallo.
3. Visitare l'array per trovare i periodi scoperti.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



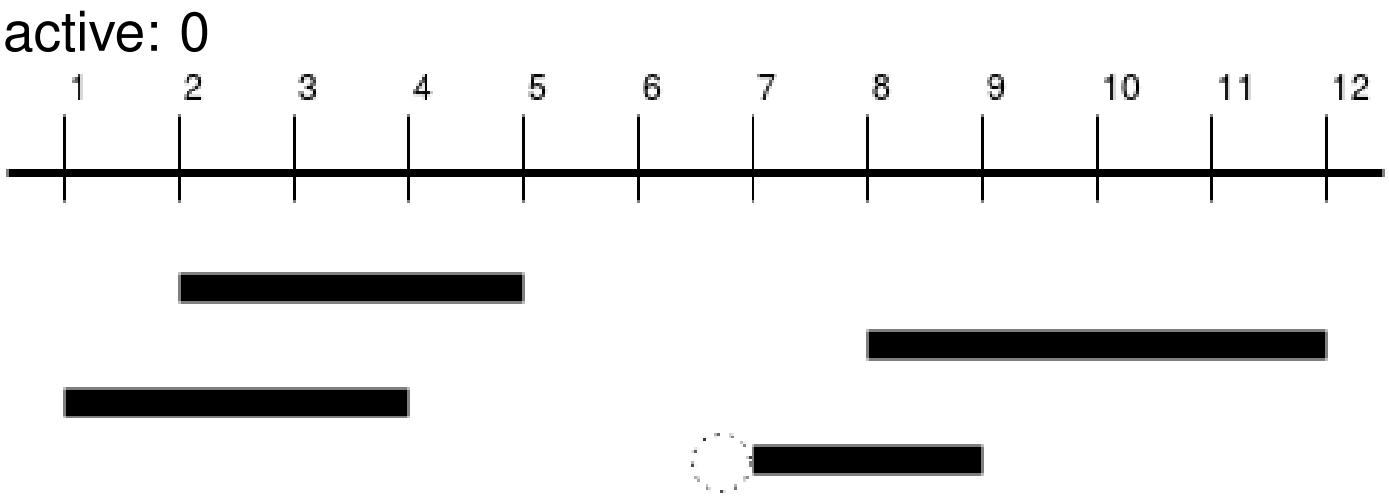
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



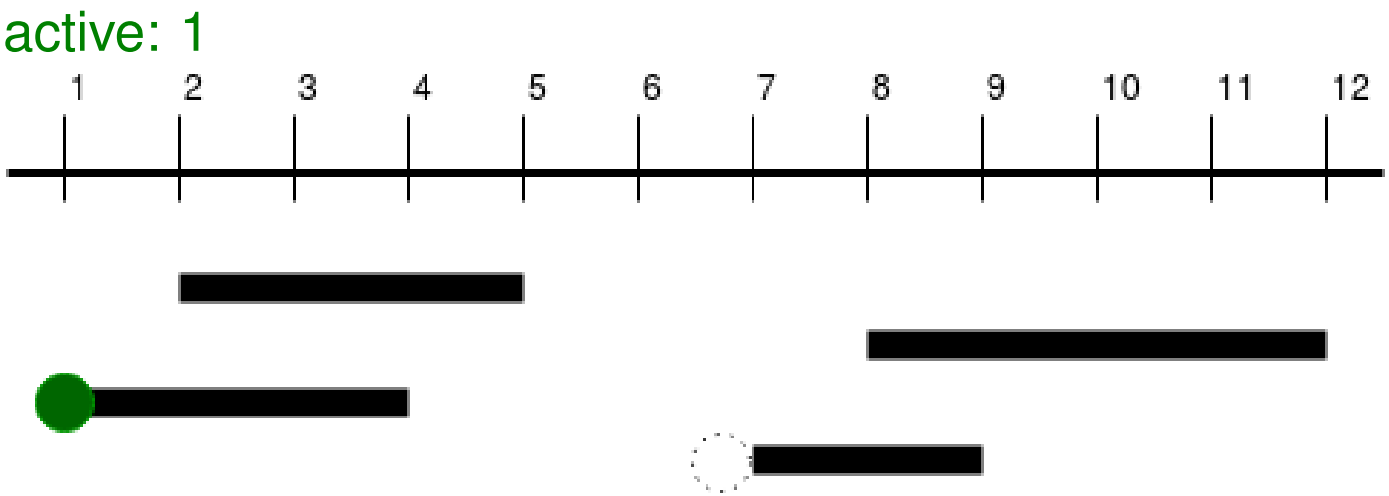
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



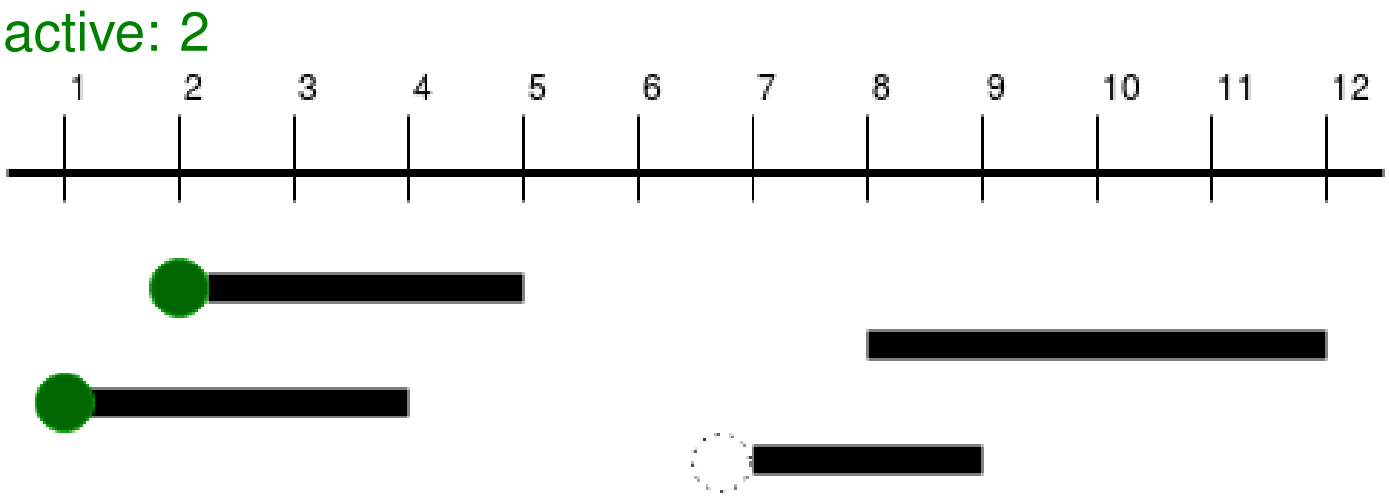
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



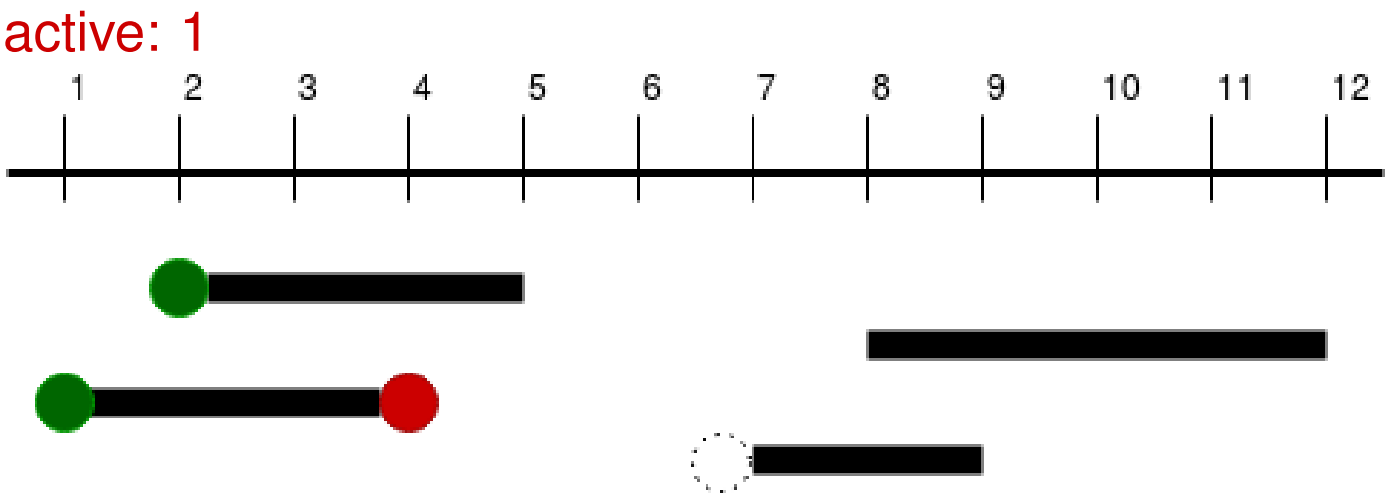
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



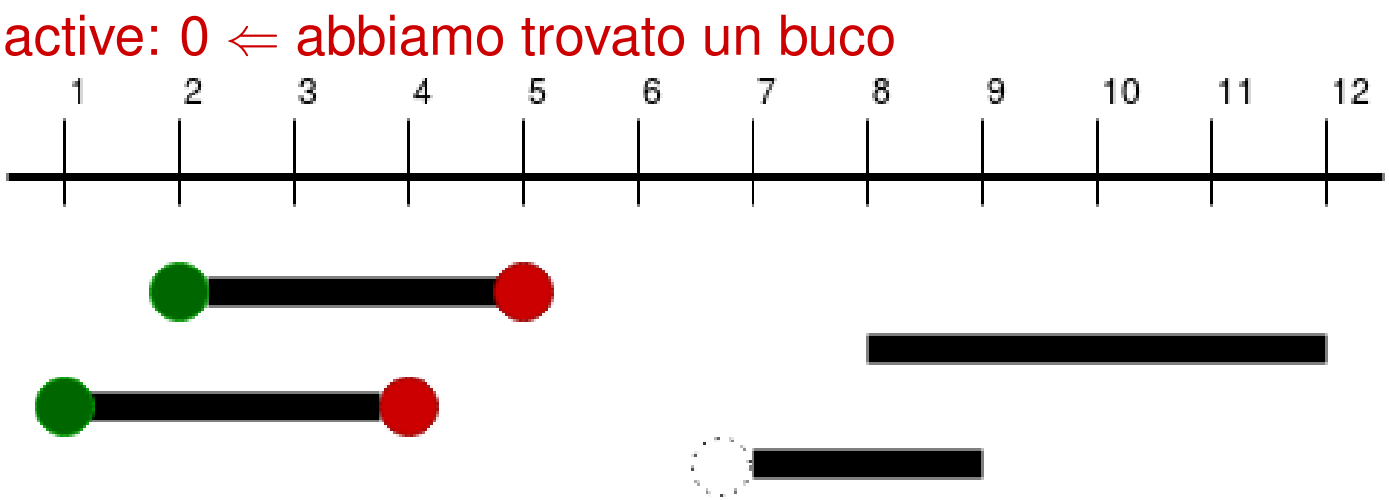
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



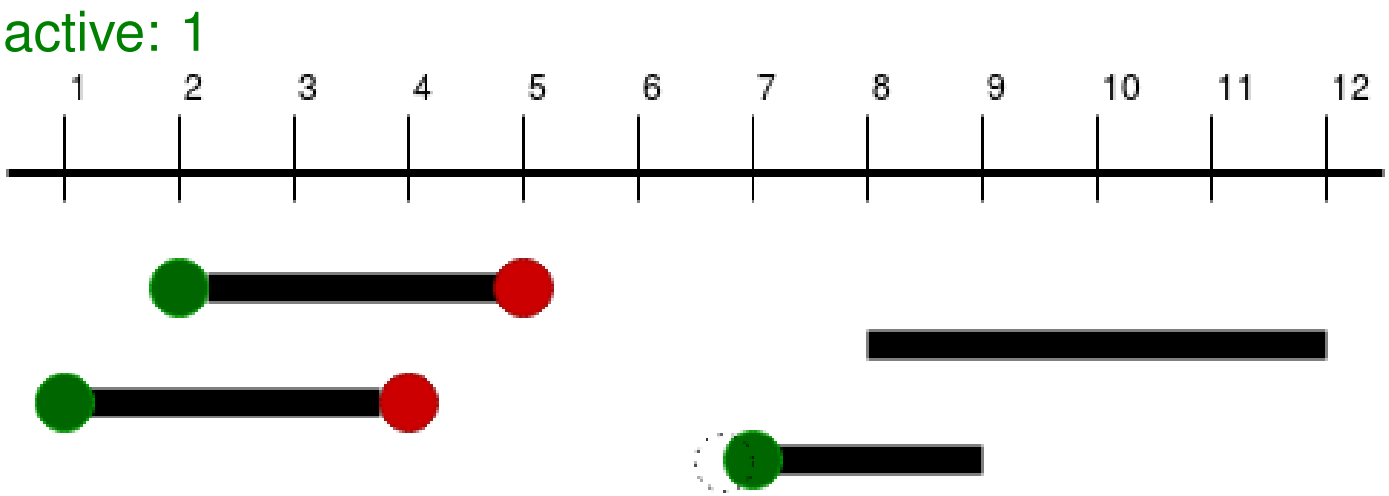
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



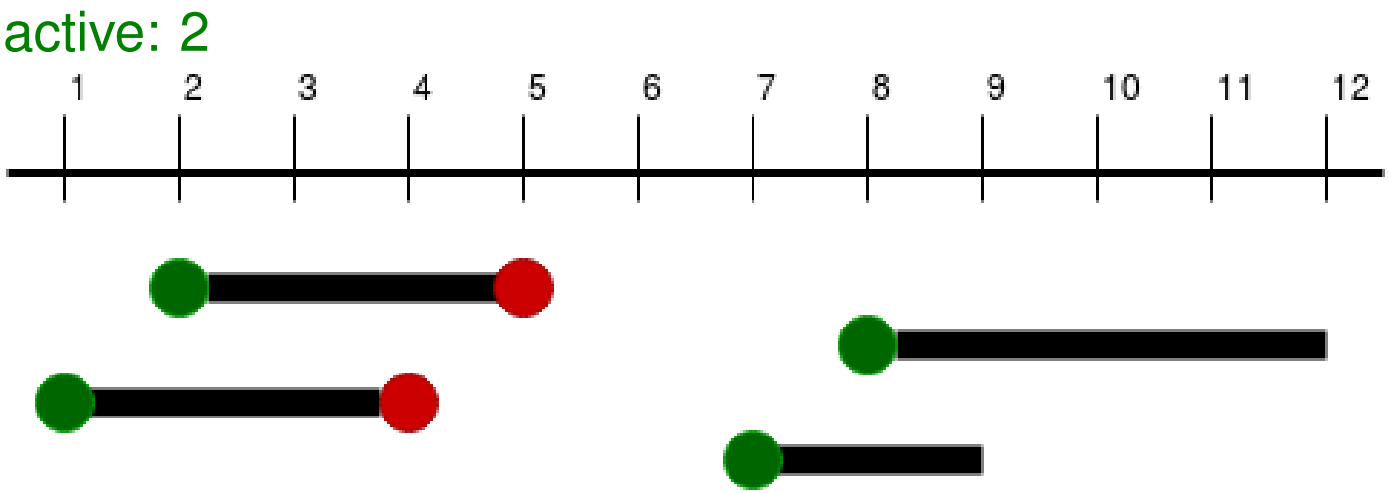
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



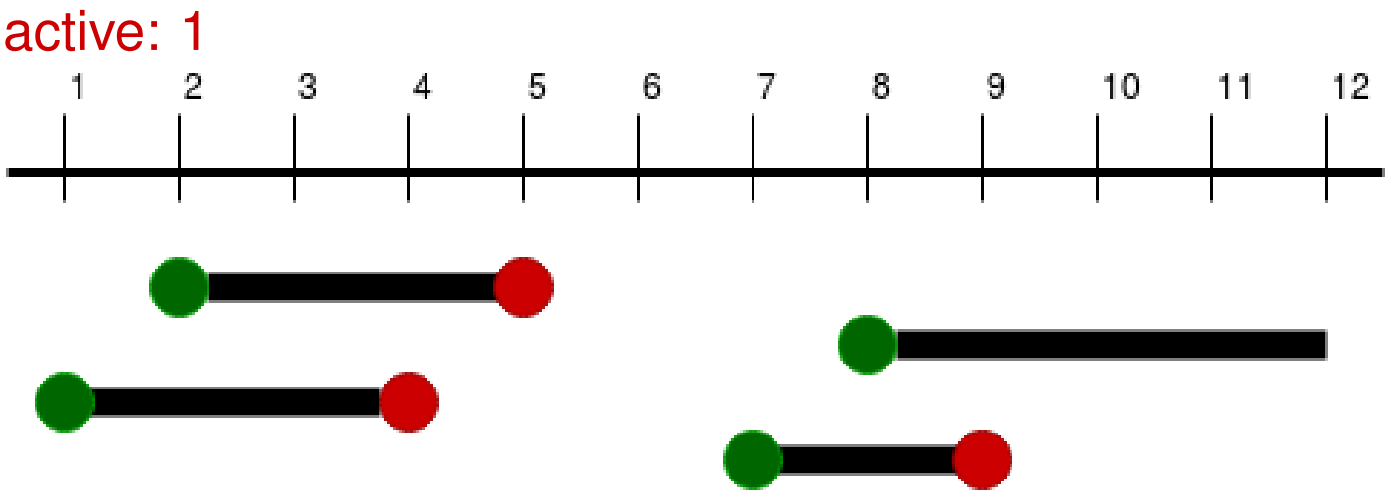
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



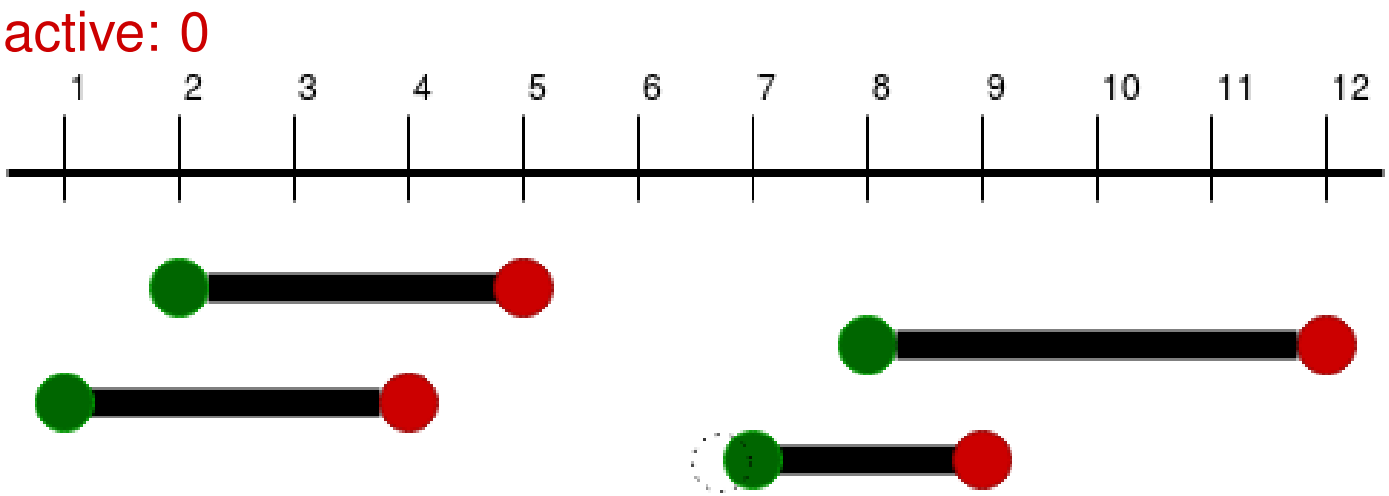
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



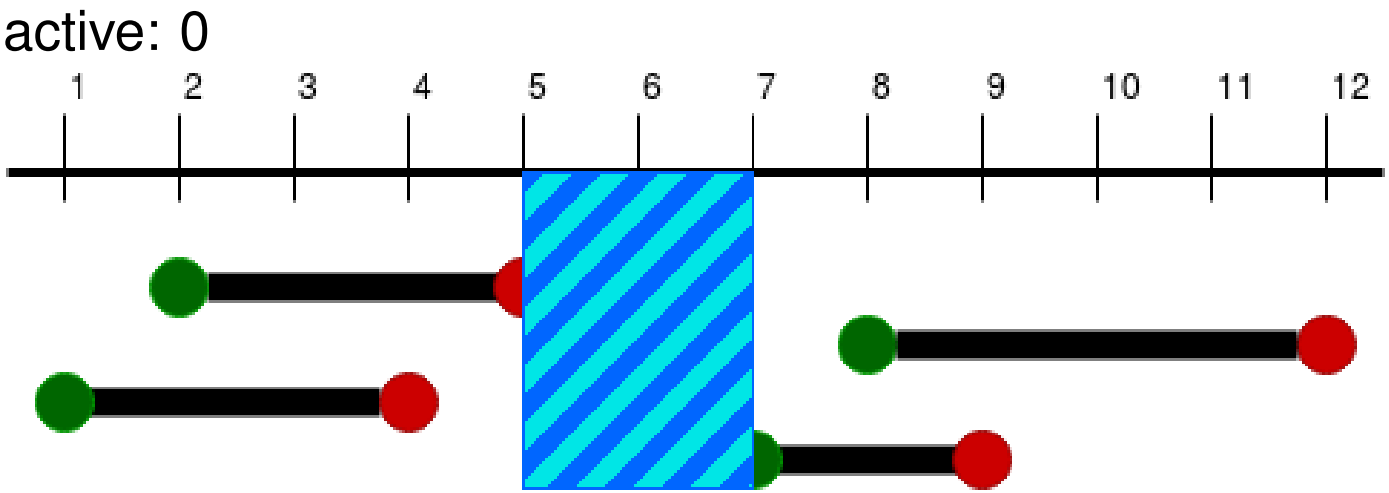
SOLUZIONI: INTERVALLO (II)

SIMULAZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Prendere tutti gli istanti di inizio e fine intervallo ed ordinarli tutti insieme, mantenendo in memoria quali si riferiscono ad un inizio intervallo e quali ad una fine intervallo.
2. Analizzare questi istanti mantenendo un contatore con il numero di intervalli attivi.
3. Se il contatore è uguale a 0, abbiamo trovato un buco.

Esempio:

```
4
2 5
8 12
1 4
7 9
```



SOLUZIONI: INTERVALLO (III)

SOLUZIONE EFFICENTE $\mathcal{O}(N \log(N))$

1. Ordiniamo gli intervalli per istante di inizio
2. L'intervallo i termina un periodo scoperto se $start[i] > end[j]$ per ogni $j < i$
3. Visitiamo gli intervalli in ordine mantenendo in memoria il massimo degli istanti di fine.

Algorithm 1 LongestHole

```
    // intervals può anche essere un const vector<pair<int,int>>&
 1: function LongestHole(intervals[], N)
        // intervals è ordinato per istanti di inizio
 2:     R ← intervals[1].end
 3:     s_h ← −1                                          // Inizio del buco
 4:     e_h ← −1                                          // Fine del buco
 5:     for i = 2 → N do                                  // N = intervals.size()
 6:         if (intervals[i].start > R) ∧ (R − intervals[i].start > e_h − s_h) then
                // Periodo scoperto da R a intervals[i].start,
                // più lungo del migliore
 7:             s_h ← R
 8:             e_h ← intervals[i].start
 9:         end if
10:         E ← max(E, intervals[i].end)
11:     end for
12:     return (s_h, e_h)
13: end function
```

Ordinamento pesato: sortpesato (I)

Dato un array di N interi da ordinare. Gli elementi sono precisamente tutti gli interi fra 1 e N. Ad ogni turno potete scambiare due elementi a scelta dell'array. Per fare ciò, pagate un prezzo pari alla somma dei due elementi.

Esempio

Per scambiare di posto l'elemento 3 e l'elemento 4 impiegate un turno e pagate 7.

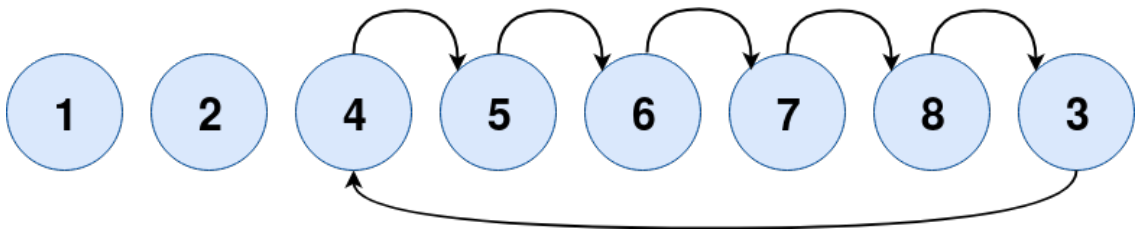
Due problemi:

- individuare il metodo più veloce, che ottimizza il numero di turni;
- individuare il metodo piú economico, che ottimizza il prezzo;

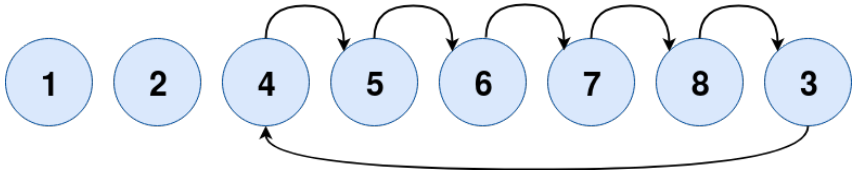
Soluzione Sort Pesato: sortpesato (I)

- Ogni elemento deve finire in una certa posizione
- Possiamo vederli come archi in un grafo da n nodi e n archi
- Individuiamo i cicli e valutiamo separatamente.
  - Il numero minimo di mosse per un ciclo di n elementi è $n - 1$
  - La soluzione ovvia sposta sempre l'elemento più piccolo con gli altri elementi
  - Può convenire prendere in prestito l'elemento 1, scambiandolo con l'elemento più piccolo del ciclo, ordinare il resto del ciclo e rimettere a posto l'elemento 1.

| 1 | 2 | 4 | 5 | 6 | 7 | 8 | 3 |



Soluzione Sort Pesato: sortpesato (II)



Possibili sequenze di scambi:

- scambiando l'elemento più piccolo, m:

$$\underbrace{(3+8)+(3+7)+(3+6)+(3+5)+(3+4)}_{n-1 \text{ scambi}} = 45$$

$\rightarrow C_m = (n-1)\cdot m + S - m$

- scambiando con 1:

$$\begin{aligned} & (1+3) \\ & + \underbrace{(1+8)+(1+7)+(1+6)+(1+5)+(1+4)}_{n-1 \text{ scambi}} \\ & + (3+1) \\ & = 43 \end{aligned}$$

$\rightarrow C_1 = 2\cdot(1+m) + (n-1) + S - m$

nota: $S = \sum_i n_i$, in questo esempio $[m = 3, n = 6, S = 33]$

Graphs 101

Dato un grafo **diretto** $G(V, E)$, esso può essere rappresentato in vari modi[1]:

1. Liste di adiacenza: per ogni nodo $v \in V$, lista dei vicini;
2. Matrice di adiacenza: $M[v][w] = 1$ se esiste l'arco $(v, w) \in E$, 0 altrimenti;
3. Lista di archi: lista degli archi $(v, w) \in E$

$\Rightarrow$ se il grafo è **indiretto** per ogni (v, w) aggiungiamo anche (w, v).

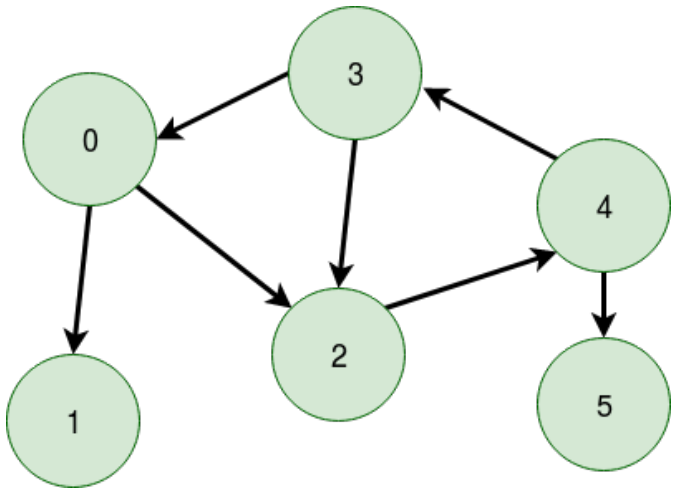
Esempi scaricabili dal sito: `https://judge.science.unitn.it/slides/asd18/graphs.zip`.

[1] (link di approfondimento)

Formato di input: esempio (I)

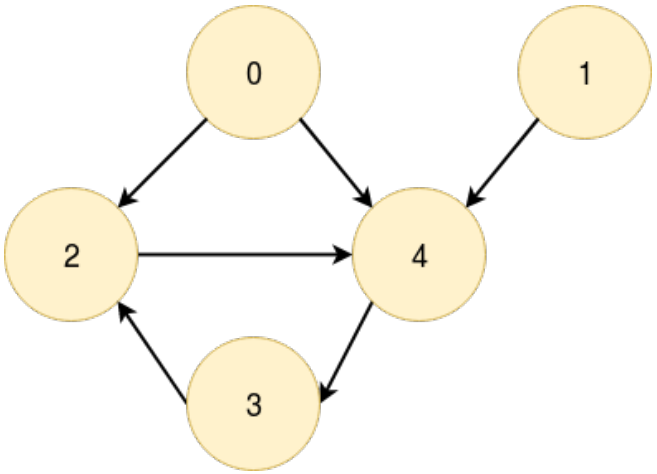
- `riga 1`: due interi N e M, rispettivamente il numero di nodi e di archi;
- `righe 2 ...M`: due interi s_i, t_i che denotano l'arco (s_i, t_i). Eventualmente possono esserci altri attributi, per esempio un peso w_i;

```
6 7
0 1
0 2
4 3
4 5
3 2
2 4
3 0
```

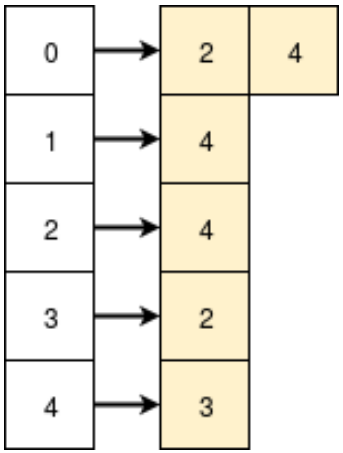


Formato di input: esempio (II)

```
5 6
0 2
0 4
1 4
3 2
2 4
4 3
```



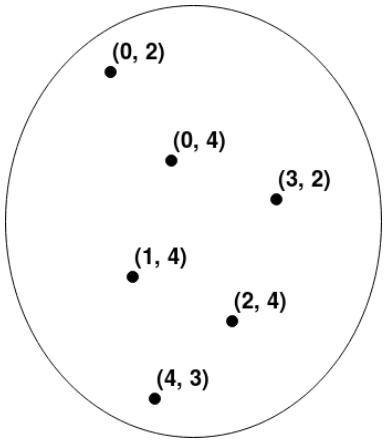
Liste di adiacenza:



Matrice di adiacenza:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Lista di archi:



Liste di adiacenza

Array di vector

Lista di adiacenza implementata come array di vector o vector di vector. Se servono altre variabili (open, close) si creano altri array

```
vector< vector <int> > adj;
vector<bool> visitato;
```

Array di struct

Struct nodo contenente lista di adiacenza e altre informazioni.

```
struct nodo{
 vector<int> vic;
 bool visitato= false;
};
```

Liste di adiacenza

```
struct nodo{
  vector<int> vic;
  bool visitato= false;
};
vector<nodo> grafo;
...
in >> N;
grafo.resize(N);
for(int i=0;i<M;i++){
  int from, to;
  in >> from >> to;
  grafo[from].vic.push_back(to);
}
```

Problemi

Visita di grafo orientato

Dato un grafo ed un nodo di partenza, trovare il numero di nodi raggiungibili a partire da quel nodo

Diametro su grafo non orientato

Dato un grafo non orientato trovare il cammino minimo di lunghezza massima tra ogni coppia di nodi del grafo, ovvero il più lungo percorso minimo fra due nodi.

Numero di cammini minimi

Dato un grafo orientato e due nodi, contare il numero di diversi cammini minimi fra i due nodi.

Problemi

007: Pioggia di Laser

Primo progetto a. a. 2017/2018

Pokémon

Primo progetto a. a. 2016/2017