Esercizio 1 Si consideri il seguente problema. Un progetto è caratterizzato da una terna (i, t_i, c_i), dove i è l'indice del progetto mentre $t_i > 0$ e $c_i > 0$ sono rispettivamente il tempo ed il costo richiesti dall'esecuzione del progetto. Un progetto è "accettabile" se non si dispone di un altro progetto che richiede tempo e costo inferiori. Dato un insieme di n progetti, con tempi e costi differenti, si vuole il sottoinsieme dei progetti "accettabili".
Ad esempio: dati i 4 progetti $\{(1,5,100),(2,7,90),(3,6,80),(4,9,110)\}$ la soluzione è $\{(1,5,100),(3,6,80)\}$.

(max 15 punti) Descrivere in pseudo-codice un algoritmo che risolve il problema in tempo $O(n \cdot \log n)$ e provarne la correttezza.

Soluzione Esercizio 1

Per quanto detto, un progetto i in P è "accettabile" se e solo se tutti i progetti di P con tempi d'esecuzione inferiori a t_i hanno un costo superiore a c_i. Una ovvia strategia consiste quindi nel prendere in esame i progetti per tempi di esecuzione crescenti e inserire un progetto nella soluzione solo se è il primo della lista oppure ha costo inferiore al costo di tutti i progetti gia considerati (quest'ultima condizione significa avere il costo inferiore al costo dell'ultimo progetto inserito nella soluzione).
Si propone pertanto il seguente algoritmo

PROGETTI : INPUT l'insieme dei progetti $P = \{(t_1, c_1), (t_2, c_2), \ldots, (t_n, c_n)\}$
  $SOL \leftarrow \emptyset$
  $c \leftarrow +\infty$
  ordina i progetti di P per tempo di esecuzione crescente.
  WHILE $P \neq \emptyset$ DO
    sia k l'indice del progetto in P che ha tempo di esecuzione minimo
    IF $c_k < c$ THEN $SOL \leftarrow SOL \cup \{k, t_k, c_k\}$
    $c \leftarrow c_k$
    $P \leftarrow P - \{k, t_k, c_k\}$
  ENDWHILE
OUTPUT SOL.

Se per ordinare l'insieme P utilizzo un algoritmo d'ordinamento efficiente la complessità dell'algoritmo proposto risulta $O(n \cdot \log n) + O(n) = O(n \cdot \log n)$.

Esercizio 2 *(max 15 punti)* Scrivere in pseudo–codice una procedura che, preso in input un grafo diretto $G = (\{1, 2, \dots n\}, E)$, due suoi vertici u e u, con $u \neq v$, ed un intero k, con $1 \leq k < n$, stampa tutti i cammini di G che vanno dal vertice u al vertice v attraversando esattamente k vertici distinti.
Ad esempio, se G è il grafo **completo** di 5 nodi, $u = 2$, $v = 3$ e $k = 2$, la procedura deve stampare (non necessariamente in quest'ordine) $(2, 1, 4, 3), (2, 1, 5, 3), (2, 4, 1, 3), (2, 4, 5, 3), (2, 5, 1, 3), (2, 5, 4, 3)$.

Soluzione Esercizio 2 Un algoritmo che risolve il problema e che si basa sulla tecnica del backtracking è il seguente. L'algoritmo ABC, essendo ricorsivo, prende in input il grafo G, gli interi n, u, v e k, il vettore SOL in cui viene memorizzato il cammino da stampare, il numero i di vertici gia toccati dal cammino fino a questo momento e il vettore caratteristico V dei vertici gia inseriti nel cammino (in pratica $V[j] = 1$ se e solo se il vertice j è stato toccato dal cammino che si sta costruendo) Quindi la prima chiamata sarà ABC(G, n, u, v, k, SOL, 0, V) dove i valori di V sono tutti a 0.

```
ABC:  INPUT il grafo G, gli interi n, u, v e k, il vettore SOL, l' intero i e il vettore V
   IF i = k AND {SOL[k], v} ∈ E THEN stampa la sequenza di vertici SOL[1], SOL[2], ..., SOL[k]
                                        preceduta dal vertice u e seguita dal vertice v
   ELSE
      IF i = 0 THEN prec ← u
      ELSE prec ← S[i]
      FOR j ← 1 TO n DO
         IF j ≠ u AND j ≠ v AND V[j] = 0 AND {prec, j} ∈ E THEN
            SOL[i + 1] ← j
            V[j] ← 1
            ABC(G, n, u, v, k, SOL, i + 1, V)
            V[j] ← 0
         ENDIF
      ENDFOR
   ENDIF
```

Sostanzialmente l'istruzione

$$\text{IF } j \neq u \text{ AND } j \neq v \text{ AND } V[j] = 0 \text{ AND } \{prec, j\} \in E \text{ THEN}$$

assicura in forma compatta che un vertice viene inserito nel cammino che si sta costruendo solo se non è uno degli estremi del cammino, non è gia stato toccato dal cammino ed in G c'è un arco che lo collega al vertice precedente del cammino.
Questa funzione di taglio non ci assicura che il cammino che si va via via costruendo può poi effettivamente completarsi in un cammino che porta al vertice v attraversando esattamente k vertici, prima di stampare bisogna quindi controllare che $(SOL[k], v)$ sia un arco in E.