Algoritmica e Laboratorio A
Soluzione Compito dell'8/7/2014

Esercizio: 1

```
QuickSort(A, 0, n-1);
for (i=0, i< n-2, i++) {
    a = A[i];
    k = i+1;
    l = n-1;
    while (k<l) {
        b = A[k];
        c = A[l];
        if (a+b+c == S) the return true;
          else if (a+b+c > S) then
            l = l - 1;
        else
            k = k + 1;
          }
    }
```

La complessità della soluzione proposta è determinata dal ciclo for che richiede $O(n^2)$.
È stata accettata anche la soluzione di $O(n^2 \log n)$ che ordina l'array e, per ogni possibile coppia di elementi a, b, cerca con RicercaBinaria se esiste un c=S-a-b.

Esercizio 4.

Un algoritmo che faccia tutte le prove possibili deve generare tutte le combinazioni di n elementi a 3 a 3, che corrispondono al coefficiente binomiale $\binom{n}{3} = O(n^3)$

ESERCIZIO 2

$G=(V,E)$ rappresentato con matrice di adiacenza A,

$$A[i,j] = \begin{cases} 0 & (i,j) \notin E \\ 1 & (i,j) \in E \end{cases}$$

Pesi: sono ~~rappresentati~~ memorizzati in una matrice P t.c.

$$P[i,j] = \begin{cases} +\infty & (i,j) \notin E \\ \text{peso arco}(i,j) & (i,j) \in E. \end{cases}$$

(1) la sequenza delle città sia memorizzata nell'array c.

VerificaPercorso (A, P, c, k)

```
km = 0;
for (i=0; i<n-1; i++) {
    if (A[c[i], c[i+1]] == 0) return false;
    km = km + P[c[i], c[i+1]];
    if (km > k) return false;
}
if (A[c[n-1], c[0]] == 0) return false;
km = km + P[c[n-1], c[0]];
if (km < k) return true;
else return false;
```

$$\boxed{T(n,m) = O(n)}$$

Esercizio 2: commento al punto (1)

Notare che per rappresentare tutta l'informazione del grafo basta ~~~~ la matrice dei pesi $P(i,j)$, ove

$P(i,j) = +\infty$ significa che il nodo i e j non sono adiacenti.

(2) Fissata la città di partenza s, per considerare tutti i possibili percorsi occorre generare tutte le permutazioni delle restanti n-1 città.

Per ogni permutazione si calcola la lunghezza del percorso associato utilizzando l'algoritmo Verifica Percorso.

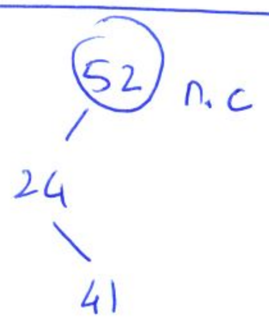
$$T(n,m) = O\left((n-1)! \cdot \underbrace{n}_{\text{costo di Verifica Percorso}}\right) = O(n!)$$

(3)

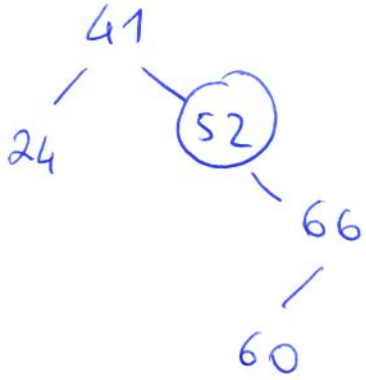
Per prima cosa, osserviamo che $TSP \in NP$. Infatti, un certificato polinomiale è dato dalla sequenza delle n città che corrispondono ad un percorso di distanza $\leq k$. Tale certificato può essere verificato in tempo polinomiale dall'algoritmo Verifica Percorso.

Dato che SAT è un problema NP completo, e $SAT \leq TSP \Rightarrow$ possiamo concludere che anche il TSP è un problema NP-completo.

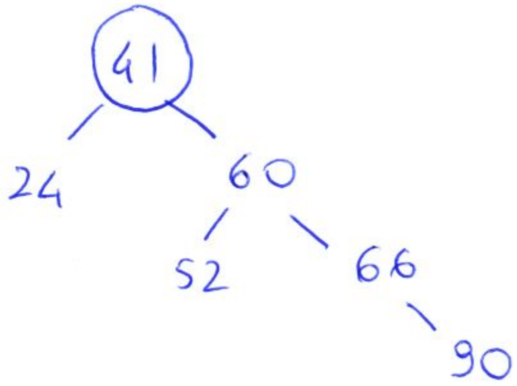
ESERCIZIO 3



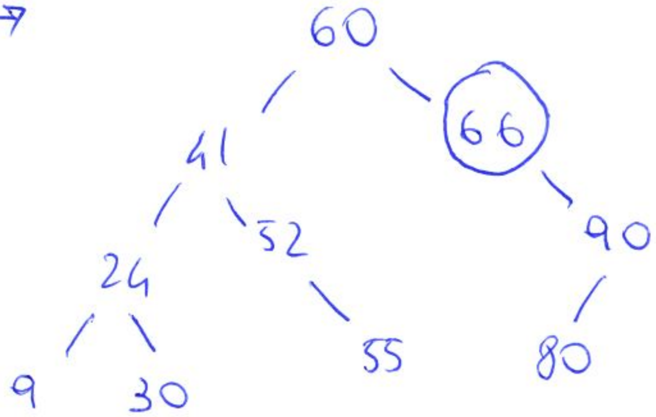
SD(52) $\longrightarrow$



DS(52) $\longrightarrow$



DD(41) $\longrightarrow$



DS(66) $\longrightarrow$