Esercizi3

Es.1

Grafo bipartito e grafo a stella

Es. 2

Triangolo

Es. 3

Algorithm 5: DynProWIS(T)

Scegli un nodo $r \in T$ come radice.
for *tutti i nodi u di T_r in post-order* **do**
    if *u è una foglia* **then**
        $\text{OPT}_{inc}(u) = w_u$
        $\text{OPT}_{esc}(u) = 0$
    else
        $\text{OPT}_{inc}(u) = w_u + \sum_{v \in children(u)} \text{OPT}_{esc}(v)$
        $\text{OPT}_{esc}(u) = \sum_{v \in children(u)} \max\{\text{OPT}_{inc}(v), \text{OPT}_{esc}(v)\}$
return $\max\{\text{OPT}_{inc}(r), \text{OPT}_{esc}(r)\}$

Es. 4

Quello a memoria

Es. 5

$ApproxVC$ può restituire l'intero grafo come output (tipo nel grafo bipartito in cui la soluzione è anche 2-approssimata).

$S*$ sol. per V.C. $k = |S|$
S sol. di $ApproxVC$ -> $k \leq |S| \leq 2k$
sia $S' = V/S$ -> $n - k \leq |S'| \leq n - 2k$

fattore di approssimazione =
Valore sol. ottima / $ApproxVC$

$(n - k)/(n - 2k) \leq 2$
$(n - k) \leq 2(n - 2k)$
$(n - k) \leq 2n - 4k$
$4k - k \leq 2n - n$
$3k \leq n$

Per cui quando il numero di nodi $n \geq 3k$ allora la soluzione di tale algoritmo è 2-approssimato.

Es. 6

L'analisi dell'algoritmo $ApproxWeightedVC$ è stretta in quanto su alcuni input la soluzione è 2-approssimata.
Indichiamo il peso di un arco con la funzione $w()$.
Sia $G = (V, E)$ un grafo bipartito t.c. $\forall e \in E, w(e) = 1$.
Metà nodi sono da una parte del grafo e i restanti nell'altra.
In questo caso ApproxWeightedVC si comporterà come ApproxVC e sarà restituito S t.c.
$|S| = 2|S*|$

Es. 7

Nel caso in cui $S = i | x_i \geq 0.75$ potrebbe non restituire proprio una solizione in quanto la soglia è troppo alta e può capitare che preso un arco $e = (u, v) \in E$ $u \notin S \wedge v \notin S$
Nel caso in cui $S = i | x_i \geq 0.25$ si va a rilassare la soglia quindi non solo avremo i valori ottenuti eseguendo l'algoritmo con soglia 0.5 ma avremo anche altri nodi che vanno addirittura a peggiorare l'approssimazione.

Es. 8

numero di macchine = n
numero task = m

es. con n=3 e m=6:
$3, 2, 1, 5, 4, 3$

Es. 9

APPROXLOAD
Siano $S = \{P_1, \ldots, P_m\}$ le macchine della soluzione ottima
Creiamo $M = \{M_1, \ldots, M_m\}$ macchine vuote
Sia $O =<>$ una sequenza vuota
while $\exists\ P_j \in S$: $P_j \neq \emptyset$:
Sia $M_i \in M$ la macchina più scarica
Sia $x \in P_i$ un lavoro
aggiungi x a M_i
rimuovi x da P_i

Aggiungere x a O

return O

Es. 10

a)

Capacità camion = C

Numero camion = 2

sia n il numero si pacchi e $0 \leq i \leq n$

se $i\%2 = 0$ -> $w_i = C/2$

se $i\%2 = 1$ -> $w_i = C/2 + 1$

b)

Sia W l'insieme dei pesi.

Sia C la capacità massima di un camion.

$B_i + B_{i+1} > C$

Supponiamo che la sol. approssimata ha usato k camion, quindi siccome la somma di due camion consecutivi è almeno C, abbiamo che:

$sum(W) = \sum_{i=1}^{|B|=k} B_i > \frac{k}{2}C > \frac{k-1}{2}C$

$\frac{k-1}{2}C$ è per il caso in cui k sia pari o dispari, questa cosa è sempre vera.

$k^\star \geq \frac{sum(W)}{C}$

$k^\star C \geq sum(W)$

$sum(W) = \sum_{i=1}^{|B|=k} B_i > \frac{k}{2}C > \frac{k-1}{2}C$

$k^{\star}C > \frac{k-1}{2}C$

$2k^{\star} > k-1$

$k-1 < 2k^{\star}$

$k \leq 2k^{\star}$

Problema di minimo : $V \leq \rho * OPT$
La soluzione è 2-approssimata.

Es. 11

```
Calcola_Molecole(M):
        while M != {}:
                Sceglia a caso x in M
                S = S+x
                M=M-{{y|d(x,y)<= ò} unito x}
        return S
```

Es.12

Dato un insieme C di città, nell'algoritmo proposto il prossimo centro verrà preso sulla circonferenza del centro precedente dato $r*$.
Ciò significa che sia $c \in C$ un centro, allora $d(c) \leq r*$ ovvero $\exists c'$ t.c. $dist(c, c') \leq r*$.
Quindi una città viene cancellata solo quando dista al massimo r dal centro.
L'algoritmo $ApproxCenter_{conR}$ Ci vuole dire se esiste un

insieme di centri che dato $r*$ riesce a ricoprire tutte le $c \in C*$.
Nel caso ci siamo due nodi molto molto distanti, l'algoritmo con questa modifica troverà una soluzione con un sono centro ma con un r molto elevato, mentre l'esecuzione standard potrebbe fornire una soluzione con r molto più piccoli aumentando però il numero di centri.
In poche parole, con questa modifica l'algoritmo può essere peggiorato a piacimento.
Dato k centri da trovare ed $r*$, tale modifica potrebbe richiedere di aumentare $r*$ all'infinito o almeno fino a che un centro non prende almeno un altro nodo.