ESERCIZI INTEGRAZIONE ESERCITAZIONE ALGORITMI DEL 7/6/2011

1) Esercizio scheduling multiprocessore (fatto in classe)
Senza ordinare i lavori abbiamo ottenuto e dimostrato un fattore di approssimazione 2
Cosa succede se ordiniamo i lavori in ordine decrescente per durata? Supponiamo $n > m$ (n lavori, m macchine)
$O(n \log n)$ per l'ordinamento
Essendo ordinati $T^*_{(ottimo)} \geq t_m + t_{m+1} \Rightarrow T^* \geq 2t_{m+1}$ (*)

t_j = tempo ultimo lavoro assegnato alla macchina i che fornisce la soluzione.
$j \geq m+1$ (per ip. $n > m$, quindi ci deve essere una macchina con almeno 2 lavori) quindi $t_j \leq t_{m+1}$
Da $T - t_j \leq \frac{1}{m} \sum_{k=1}^{m} T_k \leq \frac{1}{m} \sum_{i=1}^{m} t_j \leq T^*$ (l'abbiamo ottenuta a lezione senza ordinamento)
e da (*)
$T = (T - t_j) + t_j \leq T^* + \frac{1}{2} T^* = \frac{3}{2} T^*$.

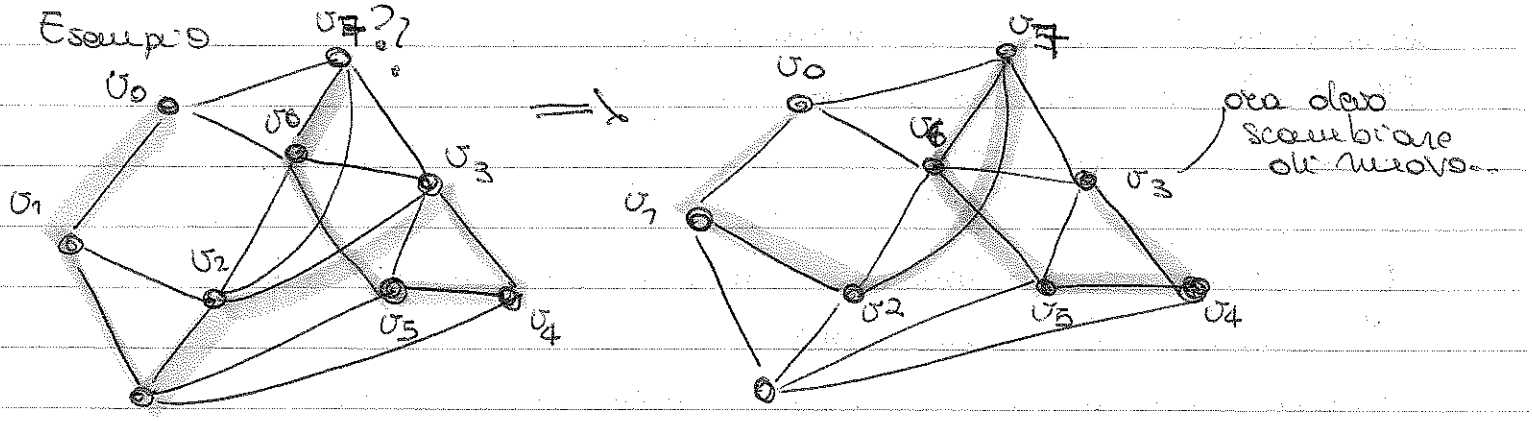
PS: utilizziamo sempre SMGREEDY visto a lezione

2) Scrivere una procedura di Las Vegas per Hamiltonian Path
$G = (V, E)$ $|V| = n$; non orientato
L'algoritmo deve non rispondere quando un cammino corretto non viene generato.
Idea: parto da v_0 arbitrario; scelgo random v_1 tra gli adiacenti di v_0; continuo fino a quando non ho coperto tutti i nodi.
Può accadere che per un certo v_i ogni v_j adiacente sia già stato considerato

Sia $v_0 v_1 \dots v_j v_{j+1} \dots v_i v_i$ il cammino costruito fino a questo momento e siano tutti gli archi di v_i già considerati.

Scambiamo il cammino con $v_0 v_1 \dots v_j v_i v_{i-1} \dots v_{j+1}$ con v_j adiacente a v_i scelto a caso.

Esempio



$v_i = v_7$, $v_j = v_2$; $v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7$ $\Rightarrow$ - - - - - + -

3) Greedy Sat approssimato

$\Phi = \{c_1, c_2 \dots c_n\}$ c_i in forma normale congiuntiva

Il seguente algoritmo è 2-approssimante ...

ovvero soddisfa almeno metà delle clausole

- Scegli il letterale che occorre il maggior n° di volte (affermato o negato); se $l = x \Rightarrow x \leftarrow v$
  se $l = \neg x \Rightarrow x \leftarrow f$
- Cancellare le clausole in cui occorre l e cancellare $\neg l$ dalle altre fino a quando $\Phi = \emptyset$.

TH: Greedy Sat è 2 approssimato.

Prova: induzione sul n° di variabili

$n = 1$ ✓

$n > 1$ Ipotesi induttiva: il risultato (2 approx) (ii) vale su $(n-1)$ variabili per m' clausole ovvero l'algoritmo mi garantisce di soddisfare $m'/2$ clausole.

Consideriamo n variabili; sia l_0 il letterale più frequente, supponiamo $l_0 = q$ che compare k_1 volte, mentre $\neg q$ compare k_2 volte (wlog $k_1 \geq k_2$)

Se $q \leftarrow v \Rightarrow$ soddisfo k_1 clausole

Quindi ora ho una formula con $(n-1)$ var. e $m' \geq m - k_1 - k_2$ clausole; per (ii) ne soddisfo $\frac{m'}{2}$

$\Rightarrow$ in tutto $k_1 + \frac{m'}{2} \geq k_1 + \frac{m - k_1 - k_2}{2} \geq \frac{m}{2}$ □

4) Branch and bound per Vertex Cover

Mi serve una funzione che marchi i nodi come selezionati o non selezionati.

Definiamo $d(v_i)$ = grado di v_i # adiacenti non marcati non è un bound

Se marco $i \Rightarrow$ riduco di $d(v_i)$ l'insieme degli archi scoperti S

Se $v_1 v_2 \dots v_k$ sono coperti $\Rightarrow$ al massimo ho ridotto l'insieme S di $d(v_1) + \dots + d(v_k)$

Se siamo ad un certo punto del backtracking e abbiamo: $|S|$ vertici ancora scoperti; k nodi; il lower bound per il minimo n° di archi da coprire nel sotto albero è

$$M = \max\left\{0, |S| - \sum_{i=1}^{k} d(v_i)\right\}$$

Non proseguo l'esplorazione del sotto albero se $M \geq opt =$ = n° vertici ottimo corrente.

Faccio backtracking quando arrivo a v_i t.c. $\forall v_j \in adj(v_i)$ v_j già marcato. Scrivere l'algoritmo in pseudocodice.

5) $G=(V,E)$, $|V|=n$, i nodi sono colorati di rosso o di nero

Ricorrenza di un algoritmo di programmazione dinamica che determini se in G esiste cammino da x a y con al più k vertici neri interni. $(x \rightsquigarrow y)$

Sol: $D(i,j,x,k) = \begin{cases} 1 & \text{se } \exists \text{ cammino da } i \text{ a } j \text{ passando per } \{1 \dots x\} \text{ e con al più } k \text{ vertici neri interni} \\ 0 & \text{altrimenti} \end{cases}$

Ad ogni step, considero x. Se x non sta sul cammino tra i e j $\Rightarrow$ riduco il problema all'istanza in cui considero come nodi intermedi $\{1 \dots x-1\}$.

Se x sta sul cammino $\Rightarrow$ divido il cammino in $i \rightsquigarrow x$ e $x \rightsquigarrow j$ con $\{1, \dots x-1\}$ come vertici intermedi per entrambi. In tutto ho al massimo k neri quindi per ogni $k \geq 0$

$$D(i,j,x,k) = \begin{cases} D(i,j,x-1,k) & \text{se } x \notin i \rightsquigarrow j \\ D(i,x,x-1,k_1), D(x,j,x-1,k_2) & \text{se } x \text{ rosso e } k_1+k_2 \leq k \\ D(i,x,x-1,k_1), D(x,j,x-1,k_2) & \text{se } x \text{ nero, } k_1+k_2+1 \leq k \end{cases}$$

Casi base: $t=0 \Rightarrow D(i,j,0,k)=1$ se $(i,j) \in E$, 0 altrimenti $\quad \forall k \geq 0$

$k=0 \Rightarrow D(i,j,k,0) = D(i,j,x-1,0)$ se $x \in i \rightsquigarrow j$

$D(i,j,x,0) = 0$ se x nero

$D(i,j,x,0) = D(i,j,x-1,0)$ se x rosso

Dov'è la soluzione?