

Batterie (Esercizio 1.3 di 14-greedy.pdf)

- Siete a bordo di un'auto elettrica su un'autostrada. Entrate in autostrada al km 0 con la batteria carica e dovete uscire al km L .
- Prima che la vostra batteria si esaurisca (dopo r km), dovete fermarvi in un'area di servizio e sostituirla con una batteria di ricambio.
- Sia $D[1 \dots n]$ un vettore di interi, dove $D[i]$ è la distanza dell'area di servizio i -esima dall'inizio dell'autostrada.
- Scrivete un algoritmo che prenda in input D , n , L e r e restituisca il numero minimo di fermate necessarie per completare il viaggio. Discutere correttezza e complessità.

Batterie (Esercizio 1.8 di 13-pd.pdf)

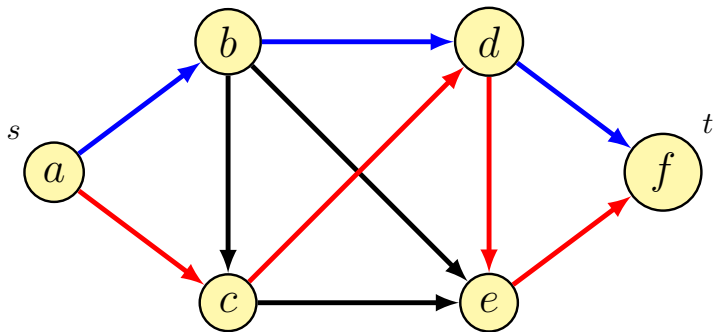
- Siete a bordo di un'auto elettrica su un'autostrada. Entrate in autostrada al km 0 con la batteria carica e dovete uscire al km L .
- Prima che la vostra batteria si esaurisca (dopo r km), dovete fermarvi in un'area di servizio e sostituirla con una batteria di ricambio.
- Siano $D[1 \dots n]$ e $C[1 \dots n]$ due vettori di interi, dove $D[i]$ è la distanza dell'area di servizio i -esima dall'inizio dell'autostrada, e $C[i]$ è il costo di una nuova batteria nell'area i .
- Il costo totale del viaggio è dato dalla somma dei costi delle batterie sostituite per arrivare al km L .
- Scrivete un algoritmo che prenda in input D e C , n , L , r e restituisca il costo totale minimo. Discutere correttezza e complessità.

Supersequenza comune minimale (Compito 10/01/2017)

- Una stringa P è una supersequenza di una stringa T se T è una sottosequenza di P .
- Scrivere un algoritmo che restituisca la lunghezza della *supersequenza comune minimale* di due stringhe P , T , ovvero la più piccola supersequenza di entrambe le stringhe.
- Discutere correttezza e complessità dell'algoritmo proposto.
- Esempio: L'unica supersequenza comune minimale di AB e BC è ABC , e la sua lunghezza è pari a 3.
- Esempio: Esistono due supersequenze comuni minimali di DAB e DCB , ovvero $DACB$ e $DCAB$, e la loro lunghezza è pari a 4.

Cammini indipendenti (Esercizio 1.5 di 15-locale.pdf)

Dato un grafo orientato $G = (V, E)$ e due vertici s, t contenuti in V , descrivere un algoritmo che restituisca il numero totale di cammini “edge-disjoint”, ovvero in cui un arco può comparire al massimo in un cammino.



Torri di controllo (Esercizio 1.7 di 15-locale.pdf)

- Si consideri un insieme di aerei $A = \{a_1, \dots, a_n\}$ e un insieme di torri di controllo $T = \{t_1, \dots, t_m\}$.
- In ogni istante, ogni aereo e ogni torre è dotato di coordinate geografiche $(a_i.x, a_i.y)$ o $(t_j.x, t_j.y)$. Ogni torre può gestire al più L aerei, e ovviamente questi devono essere a portata radio r dalla torre.
- Descrivere un algoritmo che assegni ogni aereo ad una torre, rispettando i vincoli sulla distanza e sul carico.
- Discutere correttezza e complessità.

Spoiler alert!

Batterie (Esercizio 1.3 di 14-greedy.pdf)

SET minStops(int[] D , int n , int L , int r)

int $deadline = r$

SET $stops = \text{Set}()$

for $i = 2$ to n do

 if $D[i] \geq deadline$ then
 $stops.insert(i - 1)$
 $deadline = D[i - 1] + r$

if $deadline < L$ then

$stops.insert(n)$

return $stops$

Batterie (Esercizio 1.3 di 14-greedy.pdf)

- Assumiamo che l'autostrada sia percorribile, ovvero l'autonomia r sia sufficiente per andare:
 - da una stazione alla successiva
 - dal km 0 alla prima stazione
 - dall'ultima stazione al km L
- Per minimizzare il numero di fermate, procediamo in modo greedy: cerchiamo la prima stazione che non può essere raggiunta con autonomia r e inseriamo la stazione precedente.
- Dobbiamo dimostrare:
 - Sottostruttura ottima
 - Scelta greedy
 - Correttezza nei casi limite e nei valori estremi
- La complessità dell'algoritmo è $O(n)$.

Batterie (Esercizio 1.3 di 14-greedy.pdf)

Casi limite

- La prima stazione deve essere raggiungibile con autonomia r , quindi partiamo dalla seconda
- Se l'ultima stazione a cui abbiamo fatto rifornimento non ci permette di raggiungere L , aggiungiamo l'ultima stazione;
- Se esiste una stazione sola, ci sono due casi:
 - se $r \geq L$, la carica ci permette di uscire dall'autostrada; ma allora $deadline \geq L$ e l'algoritmo restituisce l'insieme vuoto
 - se $r < L$, l'algoritmo restituisce l'unica stazione presente

Scelta greedy

- Assumiamo che esista una soluzione S che non includa la stazione k , l'ultima stazione tale che $D[k] \leq r$, a partire dall'inizio (km 0).
- sia i la prima stazione utilizzata in S , con $i < k$;
- se sostituisco i con k , ottengo una soluzione $S' = S - \{i\} \cup \{k\}$ che è comunque una soluzione ottima (ha la stessa dimensione) e rispetta tutti i vincoli (k è raggiungibile dall'inizio e può raggiungere la stazione successiva in S , perchè questa era raggiungibile da i che era più indietro).

Batterie (Esercizio 1.8 di 13-pd.pdf)

È possibile definire ricorsivamente il problema come segue. Assumiamo che esistano due stazioni fittizie $0, n + 1$, con $D[0] = 0$ e $D[n + 1] = L$ e $C[0] = C[n + 1] = 0$. Non è quindi necessario passare L .

Definiamo una tabella $DP[0 \dots n + 1]$, tale che $DP[i]$ rappresenti il costo minimo da pagare nel caso si acquisti la batteria alla stazione i -esima. La nostra soluzione si trova in $DP[0]$. $DP[i]$ può essere definito in maniera ricorsiva nel modo seguente:

$$DP[i] = \begin{cases} 0 & \text{se } i = n + 1 \\ \min_{j: j > i \wedge D[j] \leq D[i] + r} \{DP[j]\} + C[i] & \text{altrimenti} \end{cases}$$

Batterie (Esercizio 1.8 di 13-pd.pdf)

```
int minCostStops(int[]  $D$ , int[]  $C$ , int  $n$ , int  $r$ )
```

```
int[]  $DP$  = new int[ $0 \dots n + 1$ ]
```

```
 $DP[n + 1] = 0$ 
```

```
for  $i = n$  downto 0 do
```

```
    int  $j = i + 1$ 
```

```
    int  $DP[i] = +\infty$ 
```

```
    while  $j \leq n + 1$  and  $D[j] \leq D[i] + r$  do
```

```
         $DP[i] = \min(DP[i], DP[j])$ 
```

```
         $j = j + 1$ 
```

```
     $DP[i] = DP[i] + C[i]$ 
```

```
return  $DP[0]$ 
```

Batterie (Esercizio 1.8 di 13-pd.pdf)

Inviduare l'indice j per cui $D[j] \leq D[i] + r$ può richiedere $O(n)$; quindi il costo pessimo di tale algoritmo è $O(n^2)$.

Supersequenza comune minimale (Compito 10/01/2017)

La lunghezza della più lunga supersequenza comune può essere calcolata tramite la seguente espressione ricorsiva:

$$DP[i][j] = \begin{cases} i & i \geq 0 \wedge j = 0 \\ j & i = 0 \wedge j \geq 0 \\ DP[i-1][j-1] + 1 & i > 0 \wedge j > 0 \wedge p_i = t_j \\ \min\{DP[i-1][j], DP[i][j-1]\} + 1 & i > 0 \wedge j > 0 \wedge p_i \neq t_j \end{cases}$$

- Nel caso una delle stringhe sia vuota, tutti i caratteri dell'altra stringa devono essere presenti nella supersequenza, da cui i due casi base.
- Se gli ultimi caratteri delle due stringhe sono uguali, si considera il sottoproblema in cui viene rimosso l'ultimo carattere di entrambi e si aggiunge +1 per contare questo carattere.
- Nel caso siano diversi, si prenderanno i due sottoproblemi in cui si rimuove l'ultimo carattere di una delle stringhe e si aggiunge +1 per contare questo carattere, prendendo il valore più piccolo fra i due.

Supersequenza comune minimale (Compito 10/01/2017)

```
int scs(ITEM[] P, ITEM[] T, int n, int m)  
  
int[][] D = new int[0...n][0...m]  
for i = 0 to n do  
     $DP[i][0] = i$   
  
for j = 0 to m do  
     $DP[0][j] = j$   
  
for i = 1 to n do  
    for j = 1 to m do  
        if  $P[i] == T[j]$  then  
             $DP[i][j] = DP[i-1][j-1] + 1$   
        else  
             $DP[i][j] = \min(DP[i-1][j], DP[i][j-1]) + 1$   
  
return  $DP[n][m]$ 
```

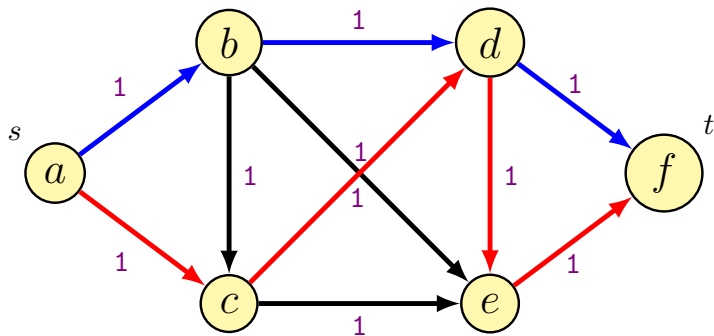
Cammini indipendenti (Esercizio 1.5 di 15-locale.pdf)

Si costruisce una funzione di capacità c tale per cui

$$c(x, y) = \begin{cases} 1 & (x, y) \in E \\ 0 & (x, y) \notin E \end{cases}$$

Si consideri il valore del flusso massimo flusso fra s (sorgente) e t (pozzo); questo valore corrisponde al numero totale di cammini indipendenti, in quanto essendo la capacità di tutti gli archi pari ad 1, ogni cammino aumentante avrà valore di flusso 1 e i suoi archi avranno capacità residua zero. In questo modo, ogni arco può essere utilizzato in un solo cammino.

Cammini indipendenti



Cammini indipendenti

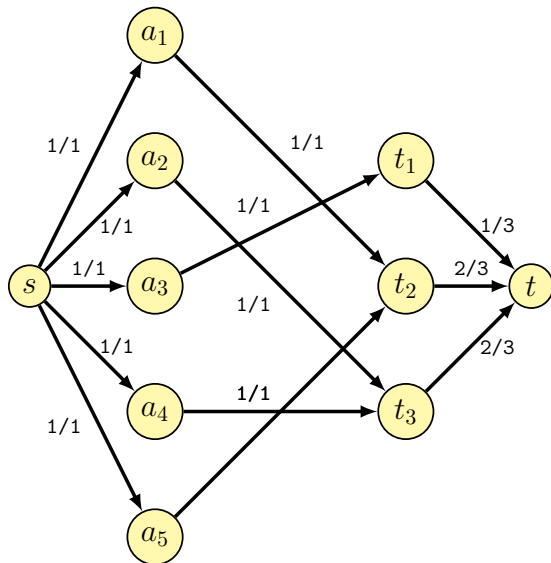
Complessità

- Seguendo il limite di Ford-Fulkerson, il costo dell'algoritmo è $O(|f^*|(n + m))$.
- $|f^*|$ è limitato da $O(n)$, perché ci sono al più $(n - 1)$ archi che escono dalla sorgente (o che entrano nel pozzo)
- Quidi la complessità risultante è $O(n(m + n)) = O(mn)$.

Torri di controllo (Esercizio 1.7 di 15-locale.pdf)

- È possibile associare ad ogni aereo ed ad ogni torre i vertici di un grafo bipartito:
- L'insieme $V = \{s, t\} \cup \{a_1, \dots, a_n\} \cup \{t_1, \dots, t_m\}$ è costituito dai nodi sorgente, pozzo, i nodi aerei e i nodi torri.
- Nell'insieme E si inseriscono:
 - un arco (s, a_i) con capacità 1 fra la sorgente e ogni aereo a_i ;
 - un arco (a_i, t_j) con capacità 1 se l'aereo a_i è entro la portata radio della torre t_j (ovvero, se $\sqrt{(a_i.x - t_j.x)^2 + (a_i.y - t_j.y)^2} \leq r$);
 - un arco (t_j, t) con capacità L fra ogni torre e il pozzo t .

Torri di controllo (Esercizio 1.7 di 15-locale.pdf)



Torri di controllo (Esercizio 1.7 di 15-locale.pdf)

- In questo modo, ogni aereo viene assegnato al massimo ad una torre di controllo; ogni torre controllo può avere assegnato fino a L aerei.
- Gli archi il cui flusso è pari a 1 corrispondono agli assegnamento aerei-torri. Il flusso massimo è limitato superiormente da n .
- Il numero di nodi è pari a $|V| = n + m + 2$;
- il numero di archi è pari a $|E| = n + m + O(nm)$.
- Utilizzando il limite derivante da Ford e Fulkerson, il costo dell'algoritmo risultante è quindi $O(n(|V| + |E|))$; il costo totale è quindi $O(n^2m)$.