Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Corso di Algoritmi e Strutture Dati

Algoritmi greedy

Algoritmi greedy

- Per alcuni problemi di ottimizzazione, un approccio greedy risulta più efficiente della programmazione dinamica
- Gli algoritmi greedy procedono top-down scegliendo di volta in volta il sottoproblema che garantisce l'ottimo "locale"
- In alcuni casi, questo modo di procedere conduce alla soluzione ottima globale
- L'approccio greedy è anche utilizzato per realizzare euristiche per la soluzione approssimata di problemi complessi (NP-hard)
- L'algoritmo di Dijkstra per il calcolo del percorso minimo utilizza un approccio greedy

Esempio: selezione di attività

- Consideriamo un insieme S di n attività che richiedono l'uso esclusivo di una risorsa comune
- Ogni attività i ha un tempo di inizio s_i e un tempo di fine f_i
  - Impegna la risorsa comune nell'intervallo $[s_i, f_i)$
- L'obiettivo è selezionare un insieme di attività mutualmente compatibili di cardinalità massima
  - a_i e a_j sono compatibili se $s_i \geq f_j$ oppure $s_j \geq f_i$

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

- Sono mutualmente compatibili
  - $\{a_3, a_9, a_{11}\}$ $\{a_1, a_4, a_8, a_{11}\}$ $\{a_2, a_4, a_9, a_{11}\}$

Esempio: selezione di attività

- Applichiamo l'approccio della programmazione dinamica
  - Poi ci accorgeremo che è possibile una semplificazione
- Per provare la sottostruttura ottima, abbiamo bisogno di identificare dei sottoproblemi
- Indichiamo con S_{ij} l'insieme delle attività compatibili con a_i e a_j

  $S_{ij} = \{a_k \in S : f_i \leq s_k \leq f_k \leq s_j\}$
- Tali attività sono anche compatibili con tutte le attività che finiscono non più tardi di a_i e tutte le attività che iniziano non prima di a_j
- Per comodità, aggiungiamo due attività fittizie a_0 e a_{n+1} tali che

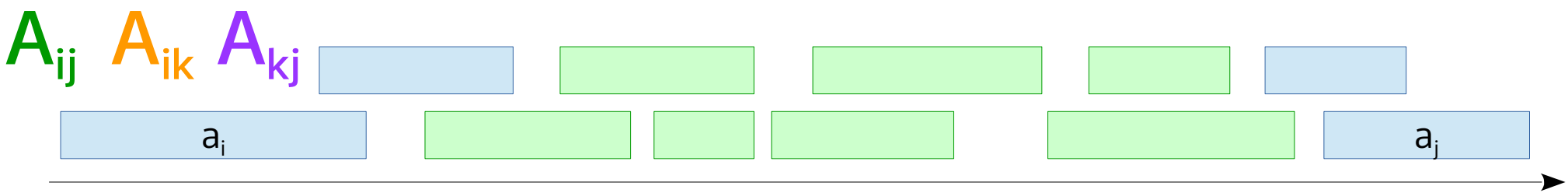
  $f_0 = 0$, $s_{n+1} = \infty$

Esempio: selezione di attività

- Supponiamo che le attività sono ordinate per tempo di fine crescente: $f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n < f_{n+1}$
- Proviamo che $S_{ij} = \emptyset$ se $i \geq j$
  - Supponiamo (per assurdo) che esista una $a_k \in S_{ij}$
  - Avremmo $f_i \leq s_k < f_k \leq s_j < f_j$
  - $f_i < f_j$ contraddice l'ipotesi che a_i segue a_j
- I sottoproblemi tra cui scegliere sono del tipo: determinare l'insieme massimo di attività mutuamente compatibili tra quelle appartenenti a S_{ij}, con $0 \leq i < j \leq n+1$ (gli altri sono vuoti)

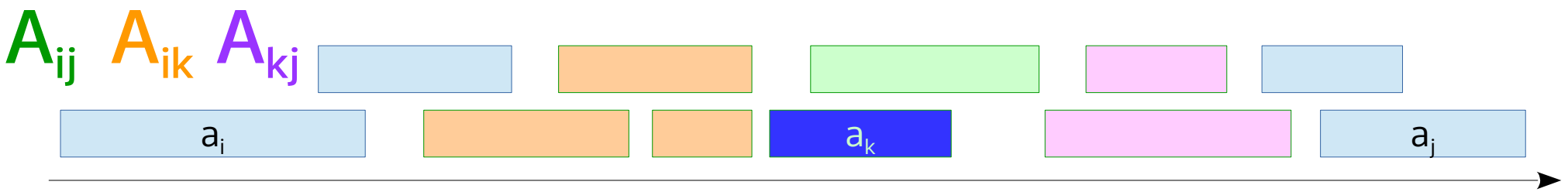
Esempio: selezione di attività

- Vediamo come la soluzione di un problema si può ottenere a partire dalla soluzione di sottoproblemi
- Sia a_k un'attività inclusa in S_{ij} (quindi $f_i \leq s_k < f_k \leq s_j$)
- Si generano due sottoproblemi
  - S_{ik} (attività che iniziano dopo che a_i finisce e finiscono prima che a_k inizi)
  - S_{kj} (attività che iniziano dopo che a_k finisce e finiscono prima che a_j inizi)
- *Una* soluzione ad S_{ij} è data dall'unione delle soluzioni a S_{ik} e S_{kj} più a_k



Esempio: selezione di attività

- La sottostruttura ottima si dimostra in questo modo
  - Supponiamo che una soluzione ottima A_{ij} di S_{ij} includa a_k
  - Le soluzioni A_{ik} di S_{ik} e A_{kj} di S_{kj} incluse in A_{ij} devono essere ottime
  - Per assurdo: se esistesse una soluzione A'_{ik} di S_{ik} con più attività di A_{ik} potremmo sostituire A'_{ik} a A_{ik} in A_{ij}, ottenendo un insieme di cardinalità maggiore rispetto alla soluzione ottima (analogo per A_{kj})



Esempio: selezione di attività

- Il passo successivo è definire il valore di una soluzione ottima in maniera ricorsiva
- Se a_k appartiene alla soluzione ottima di S_{ij}, il valore della soluzione ottima di S_{ij} sarà

  $c[i,j] = c[i,k] + c[k,j] + 1$
- Tuttavia, k non è noto, può essere una delle attività comprese tra a_{i+1} e a_{j-1}
  - $j-i-1$ scelte

$$c[i,j] = \begin{cases} 0 & \text{se } S_{ij} = \emptyset \\ \max_{i<k<j} \{c[i,k]+c[k,j]+1\} & \text{se } S_{ij} \neq \emptyset \end{cases}$$

Esempio: selezione di attività

- A questo punto, si potrebbe utilizzare l'approccio bottom-up della programmazione dinamica

  Teorema 1. Sia S_{ij} un sottoproblema non vuoto e a_m l'attività in S_{ij} con il tempo di fine più piccolo $f_m=\min\{f_k:a_k \in S_{ij}\}$. Allora

  - *L'attività a_m è usata in qualche insieme di dimensione massima di attività mutualmente compatibili in S_{ij}*
  - *Il sottoproblema S_{im} è vuoto, quindi scegliendo a_m si ha un solo sottoproblema non vuoto S_{mj}*
- Questo risultato ci consente di
  - Avere una sola scelta quando si risolve un sottoproblema (l'attività che finisce prima), facile da calcolare
  - Dover risolvere un solo sottoproblema (l'altro è vuoto)

Esempio: selezione di attività

- Intuitivamente, scegliere l'attività che finisce prima significa avere più possibilità di schedulare altre attività
  - Questa è la scelta greedy, che minimizza un ottimo "locale"
- Implementazione ricorsiva

```
Recursive-Activity-Selector (s,f,i,j)
m ← i+1
while m<j and s_m < f_i   // find the first activity in S_ij
    do m ← m + 1
if m < j
       then return {a_m} ∪ Recursive-Activity-Selector (s,f,m,j)
       else return ∅
```

- La complessità è $\Theta(n)$, dato che ogni attività è esaminata una sola volta nel ciclo while

Esempio: selezione di attività

- La procedura vista ha una ricorsione "quasi" in coda
- Le procedure ricorsive in coda sono facilmente convertibili in procedure iterative

```
Greedy-Activity-Selector (s,f)
n ← length[s]
A ← {a1}
i ← 1
for m ← 2 to n
    do if sm ≥ fi
        then A ← A ∪ {am}
             i ← m
return A
```

- Richiede un tempo $\Theta(n)$, se i valori in ingresso sono ordinati

Algoritmi greedy

- Un primo segno dell'applicabilità di un approccio greedy è la possibilità di effettuare la scelta "greedy"
  - La soluzione globalmente ottima può essere raggiunta mediante una sequenza di scelte greedy, ovvero scelte che tendono ad un ottimo locale
- Nella programmazione dinamica, la scelta tra sottoproblemi può essere effettuata *dopo* aver risolto i sottoproblemi
  - Approccio bottom-up
- Gli algoritmi greedy prima effettuano la scelta e poi risolvono il sottoproblema scelto
- Occorre però provare che la scelta greedy conduce all'ottimo globale

Algoritmi greedy

- Un altro segno dell'applicabilità di un approccio greedy (e della programmazione dinamica) è la presenza della sottostruttura ottima
- L'approccio va scelto in base alla possibilità di dimostrare che una scelta greedy conduce all'ottimo globale
- Attenzione a non
  - Utilizzare la programmazione dinamica quando l'approccio greedy è applicabile
  - Utilizzare un approccio greedy quando è richiesta la programmazione dinamica