TECNICA GREEDY: ALGORITMO DI MOORE

Questa parte non è presente in nessuno dei testi adottati.

Il problema

Dato un **insieme di lavori** (**jobs**), caratterizzati ciascuno da una **durata** e da una **scadenza**, trovare il **massimo numero** di **lavori** che **possono essere eseguiti entro la** loro **scadenza**.

Esempio: Il problema *Missioni* dalle Olimpiadi di Informatica.

Il Commissario Basettoni ha presentato a Topolino le missioni che egli dovrà svolgere segretamente nel corso dell'anno. Per ogni missione, oltre al luogo da raggiungere, Basettoni ne indica la durata in giorni e la data massima entro cui deve essere completata. In altri termini, la missione può iniziare in qualunque giorno dell'anno ma deve durare esattamente il numero di giorni indicato e terminare non oltre la data di scadenza.

Topolino, presa la lista delle missioni ricevuta da Basettoni, ordina tali missioni in base alla loro data di scadenza. Quindi, numera i giorni dell'anno da 1 a 365 (non esistono anni bisestili a Topolinia) e trasforma le date di scadenza in numeri secondo tale numerazione. Per esempio, se una missione dura 15 giorni e deve essere svolta entro il 18 febbraio, Topolino la vede semplicemente come una coppia di interi 15 49 (in quanto il 18 febbraio è il quarantanovesimo giorno dell'anno).

Esempio (continua)

Poiché può effettuare una sola missione alla volta, Topolino non sarà in grado di svolgerle tutte, pur iniziando una missione il giorno immediatamente successivo a quello in cui termina la precedente. Vuole perciò sapere il numero massimo di missioni che è in grado di eseguire rispettando i vincoli sulla loro durata e scadenza. Supponendo che Topolino già fornisca le coppie di interi ordinate per scadenza (il secondo membro delle coppie), aiutatelo a calcolare il massimo numero di missioni che può svolgere.

Per esempio, se ci sono quattro missioni, una di tre giorni da terminare entro il 5 gennaio, una di quattro giorni entro l'8 gennaio, una di tre giorni entro il 9 gennaio e una di 6 giorni entro il 12 gennaio, Topolino vi fornisce la lista di quattro coppie 3 5, 4 8, 3 9 e 6 12. Il numero massimo di missioni che può svolgere è pari a tre, ossia le missioni corrispondenti alle coppie 3 5, 3 9 e 6 12: la prima missione inizia il primo di gennaio e termina il 3 gennaio; la seconda inizia il 4 gennaio e termina il 6 gennaio; la terza inizia il 7 gennaio e termina il 12 gennaio. (Notare che, scegliendo la missione corrispondente alla coppia 4 8, Topolino può svolgere al più due missioni.)

Scheduling

Una ***sequenza di scheduling***, o brevemente uno ***scheduling***, è una sequenza di lavori $L_1, L_2, \ldots, L_n$ che devono essere eseguiti **uno dopo l'altro consecutivamente**.

Se $d_1, d_2, \ldots, d_n$ sono le durate rispettive di $L_1, L_2, \ldots, L_n$,

e $s_1, s_2, \ldots, s_n$ ne sono le rispettive scadenze,

e l'istante iniziale è 0, allora il lavoro L_i :

- inizia l'esecuzione all'istante $t_{i-1} = d_1 + d_2 + \ldots + d_{i-1}$;
- termina l'esecuzione all'istante $t_i = d_1 + d_2 + \ldots + d_{i-1} + d_i$;
- rispetta la scadenza se $t_i \leq s_i$.

Dato un insieme di job con le rispettive scadenze e durate, come trovo un sottoinsieme massimale di job (con il maggior numero di elementi) ed il suo scheduling?

Descrizione informale dell’algoritmo risolvente.

1. Ordina la sequenza dei lavori per **ordine crescente di istante di scadenza**.
2. Scandisce tale sequenza nell’ordine, aggiungendo **ogni volta il successivo lavoro alla fine di una sequenza di scheduling provvisoria**: se così facendo il lavoro considerato termina **dopo la scadenza**, si elimina dalla sequenza di scheduling (includente l’ultimo) il lavoro di **durata massima** (che così diventa un lavoro definitivamente scartato).

NOTA: l’algoritmo di Moore non rispetta alla lettera lo schema di algoritmo greedy visto precedentemente. Infatti un lavoro può essere scartato anche dopo essere stato aggiunto alla soluzione.

L’algoritmo risolvente.

1. ordina la sequenza dei lavori (job) per ordine **crescente di istante di scadenza**: **L_1, L_2, …, L_n**.
2. inizializza la sequenza-soluzione **Sol** dei job schedulati come sequenza vuota, e inizializza il tempo **t** a **0** (oppure all’istante iniziale dato).
3. **for** i **=** 1 to n

    aggiungi L_i a Sol ;

    t **=** t + durata di L_i ;

    if (t > scadenza di L_i)

        togli da Sol il lavoro L_{max} di durata massima;

        t **=** t – durata di L_{max}

Dimostrazione di correttezza: l'invariante.

Situazione al passo generico (invarianti del ciclo):

Sia **S** l'insieme di tutti i **job finora esaminati**

1. **Sol** è uno **scheduling massimale** $L_1, L_2, \ldots, L_k$ di job di S che rispetta le scadenze, cioè fra tutti gli scheduling (di job di S) che rispettano le scadenze è quello (o uno di quelli) con il **numero massimo** di elementi;
2. inoltre Sol è, fra tutti gli scheduling massimali di job di S, quello (o uno di quelli) di **durata totale minima**, dove
durata totale = durata(L_1) + durata(L_2) + … + durata(L_k)
3. Sol è **ordinato per tempi di scadenza crescenti**;
4. ogni job $L \notin S$ ha una scadenza **posteriore o uguale** alle scadenze di tutti i job $\in S$.

L'invariante è banalmente vero nel caso base, con S e Sol vuoti.

Dimostrazione di correttezza: il passo

Sia t_k l'istante di **fine** dello scheduling Sol = L_1, L_2, …, L_k .

Sia **L** il (primo) job non ancora esaminato, cioè $\notin$ S, che ha **scadenza prima** di tutti gli altri, cioè il primo job ancora da esaminare nella sequenza dei job ordinata per scadenza.

Siano **s = scadenza di L** **d = durata di L**

Consideriamo allora l'insieme **S' = S $\cup$ {L}** e cerchiamo di trovare per S' uno scheduling **massimale** di **durata minima**.

2 casi:

- L aggiunto ad S è **eseguibile** entro la (sua) scadenza
- L **non** è **eseguibile**

Dimostrazione del passo: caso 1

caso 1) $\mathbf{t_k + d \leq s}$: cioè L, aggiunto al fondo di Sol, **risulta eseguibile entro la scadenza**

Allora lo scheduling che così si ottiene $\mathbf{L_1, L_2, \ldots, L_k , L}$ è uno **scheduling massimale** per $S \cup \{L\}$.
DIMOSTRAZIONE. $\{L\}$ ha cardinalità 1. Quindi, se vi fosse uno scheduling per $S \cup \{L\}$ con più di k+1 elementi, vi sarebbe uno scheduling per S con più di k elementi (assurdo).

Inoltre $S \cup \{L\}$ è di **durata minima**,
DIMOSTRAZIONE. L è l'unico elemento che ho aggiunto e ha durata costante. Quindi se esistesse uno scheduling di durata minore, ce ne sarebbe uno di durata minore anche per S (assurdo).

Dimostrazione del passo: caso 2

caso 2) **$t_k + d > s$** : cioè L, se schedulato al fondo di Sol, **non rispetta la scadenza**;

quindi lo scheduling **$L_1, L_2, \ldots, L_k$, L non è una soluzione**.

Anche scambiando L con uno qualunque dei job precedenti L_j si ottiene uno scheduling **$L_1, \ldots, L_{j-1}, L, L_{j+1}, \ldots, L_{k-1}, L_j$** che **non rispetta tutte le scadenze**:

il suo istante finale è infatti ancora **$t_k + d$**; ma per il punto 4 dell'invariante la scadenza s di L è posteriore alle scadenze di tutti i job di S e quindi di Sol, allora

$$t_k + d > s > s_j$$

il job L_j non rispetta la scadenza.

Consideriamo due casi:

caso 2.1) **$d \geq$ massima delle durate di $L_1, L_2, \ldots, L_k$** ;

caso 2.2) **$d <$ massima delle durate di $L_1, L_2, \ldots, L_k$** .

Dimostrazione del passo: caso 2

caso 2.1) **$d \geq$ massima delle durate di $L_1, L_2, \ldots, L_k$ (L è il processo con durata massima):**

se si sostituisce un L_i con L si ottiene per $S \cup \{L\}$ uno scheduling di k elementi, di **durata totale maggiore o uguale**;

quindi scartando L, **$L_1, L_2, \ldots, L_k$ è uno scheduling massimale di durata minima anche per $S \cup \{L\}$** .

caso 2.2) **$d <$ massima delle durate di $L_1, L_2, \ldots, L_k$ (esiste L_{max} di durata maggiore di L):**

eliminando il job L_{max} di durata massima e aggiungendo al fondo il job L si ottiene per $S \cup \{L\}$ uno scheduling ancora di **k elementi**, ma evidentemente di **durata minore**, anzi di durata minima (si è "sostituito" con L il più lungo degli L_i):

$L_1, \ldots, L_{max-1}, L_{max+1}, \ldots, L_k$, L è uno scheduling massimale di durata minima per $S \cup \{L\}$

Fine della dimostrazione

Abbiamo quindi dimostrato che in tutti i possibili casi gli invarianti

L_1, L_2, …, L_k è uno scheduling massimale

Tra tutti gli scheduling massimali L_1, L_2, …, L_k è quello di durata minima

Sono mantenuti dall’algoritmo, di conseguenza sono veri **alla fine dell’esecuzione**. CVD

Cosa devo aver capito fino ad ora

- Il problema dello scheduling di processi
- L’algoritmo di Moore per la risoluzione del problema di scheduling
- La correttezza dell’algoritmo di Moore

…se non ho capito qualcosa

- Alzo la mano e chiedo
- Ripasso sul libro
- Chiedo aiuto sul forum
- Chiedo o mando una mail al docente