Olimpiadi di Informatica 2011
Giornate preparatorie

Dipartimento di Informatica
Università di Torino

marzo 2011

8 – Algoritmi greedy: l'algoritmo di Moore.
(versione 04/04/11)

Il problema: scheduling di lavori per una CPU.

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 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.)

Nota

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 $t_i \leq s_i$.

Descrizione informale dell'algoritmo risolvente.

- Ordina la sequenza dei lavori per ordine crescente di istante di scadenza.
- Scandisci 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).

L'algoritmo risolvente.

- ordina la sequenza dei lavori (jobs) per ordine crescente di istante di scadenza: $L_1, L_2, \ldots, L_n$.
- inizializza la sequenza-soluzione Sol dei jobs schedulati come sequenza vuota, e inizializza il tempo t a 0 (oppure all'istante iniziale dato).
- 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_{imax} di durata massima;
  t = t - durata di L_{imax}

Nota

Se (come nel problema *Missioni*) è richiesto solo il numero massimo di lavori schedulabili e non la loro sequenza, si può realizzare l'estrazione del lavoro di durata massima in tempo logaritmico invece che lineare, implementando Sol come heap a massimo (coda con priorità a massimo).

Ricorda, però, che in questo modo l'inserimento in Sol è anch'esso in tempo logaritmico anziché costante.

Dimostrazione di correttezza: l'invariante.

Situazione al passo generico (invariante del ciclo):

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

Dimostrazione di correttezza: il passo.

- Sia t_k l'istante di fine dello scheduling Sol = $L_1, L_2, \ldots, L_k$.
- Sia L il 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 jobs ordinata per scadenza.
- Siano s = scadenza di L d = durata di L
- caso 1) $t_k + d \leq s$: cioè L, aggiunto al fondo di Sol, risulta eseguibile entro la scadenza; allora lo scheduling che così si ottiene: $L_1, L_2, \ldots, L_k, L$
  è uno scheduling massimale per $S \cup \{L\}$, poiché 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;
  inoltre è di durata minima, perché se ve ne fosse uno di durata minore, ce ne sarebbe uno di durata minore anche per S.

Dimostrazione del passo: caso 2 e conclusione.

- caso 2) $t_k + d > s$: cioè L, se schedulato al fondo di Sol, non rispetta la scadenza. Quindi non posso semplicemente aggiungerlo alla sequenza; ma se fra gli $L_1, L_2, \ldots, L_k$ già scelti ce n'è uno L_i di durata d_i maggiore della durata d di L, conviene togliere quello e inserire L;
  anzi, conviene togliere l'elemento L_i di durata massima.
  Ossia, per rendere il codice più semplice:
  - aggiungo comunque L al fondo di $L_1, L_2, \ldots, L_k$;
  - tolgo dallo scheduling $L_1, L_2, \ldots, L_k, L$ il job di durata massima.

In conclusione, il trattamento dei due casi è semplicemente:

- aggiungo L al fondo di $L_1, L_2, \ldots, L_k$;
- se in questo modo L termina dopo la scadenza, tolgo dallo scheduling $L_1, L_2, \ldots, L_k, L$ il job di durata massima.