Laboratorio di Algoritmi e Strutture Dati

Esercitazioni del'11 Dicembre 2012

Esercizio 2: catena di montaggio

L'esercizio è diviso in due parti.
La prima parte verte sull'allocazione dinamica di array (`calloc`).
La seconda parte è un esercizio di *programmazione dinamica*.

Prima parte

In una fabbrica un certo prodotto P viene costruito attraverso n stadi di lavorazione. La fabbrica dispone di due distinte linee per la produzione di P. Ognuna delle linee è caratterizzata da una n-pla di valori interi non negativi che rappresentano il costo di produzione associato ad ognuno degli n stadi di lavorazione. In particolare:

- Alla linea A è associata la n-pla di interi $\langle a_0, \ldots, a_{n-1} \rangle$ e alla linea B la n-pla di interi $\langle b_0, \ldots, b_{n-1} \rangle$.
- Per ogni $i = 0, \ldots, n-1$, il valore a_i rappresenta il costo di produzione dell'i-esimo stadio sulla linea A; analogamente, b_i rappresenta il costo di produzione dell'i-esimo stadio sulla linea B.

Si vuole ottimizzare il costo di produzione di P, sfruttando la possibilità di scegliere, per ogni stadio di lavorazione, la linea più conveniente tra A e B.
In dettaglio: sia X una n-pla $\langle x_0, \ldots, x_{n-1} \rangle$ tale che: $x_i = A$ o $x_i = B$, per ogni $i = 0, \ldots, n-1$ (*implementativamente*, si usi $x_i = 0$ per modellare $x_i = A$ e $x_i = 1$ per modellare $x_i = B$).
Sia $c_i = a_i$ se $x_i = A$, $c_i = b_i$ se $x_i = B$. Il costo totale $C(X)$ di X è dato da

$$\sum_{i=0}^{n-1} c_i \, .$$

Si deve calcolare una n-pla X tale che il Il costo $C(X)$ sia minimo tra tutte le possibili scelte.

I costi di lavorazione delle due catene sono contenuti in un file di dati nel seguente formato:

```
n
a_0 a_1 ... a_{n-1}
b_0 b_1 ... b_{n-1}
```

Il programma deve:

1. Chiedere all'utente il nome del file dati (un file `.txt`) da cui leggere i costi di lavorazione.
2. Leggere i dati dal file specificato e caricarli in due array `a` e `b` di n elementi ciascuno (oppure in un unico array di $2n$ elementi), allocati (allocato) dinamicamente. (Si suppone che il formato del file sia sempre corretto).
3. Calcolare una n-pla $\langle x_0, \ldots, x_{n-1} \rangle$ di costo minimo.
4. Stampare la soluzione associata a tale n-pla nel formato esemplificato qui di seguito.

Esempio Si supponga che il file di input sia:

```
7
2 7 9 1 4 3 1
5 1 4 3 7 8 2
```

Allora una 7-pla di costo minimo è $\langle A, B, B, A, A, A, A\rangle$: l'output va visualizzato come segue:

```
Costo totale: 16

 2  --  --   1   4   3   1
--   1   4  --  --  --  --
```

Seconda parte

Anche in questo esercizio, si vuole ottimizzare il costo di produzione di P, sfruttando la possibilità di scegliere, per ogni stadio di lavorazione, la linea più conveniente tra A e B.
Ma in questo esercizio, ogni volta che si cambia linea di lavorazione si deve pagare un *costo di trasferimento* aggiuntivo.
In particolare le due linee sono caratterizzate da due ulteriori $(n-1)$-ple di interi non-negativi:
$\langle ab_0, ab_1, \ldots, ab_{n-2}\rangle$ e $\langle ba_0, ba_1, \ldots, ba_{n-2}\rangle$,
da interpretare come segue:

- Per ogni $i = 0, \ldots, n-2$, se lo stadio di lavorazione i avviene sulla linea A e lo stadio di lavorazione $i+1$ avviene sulla linea B, allora al costo precedentemente computato si deve aggiungere il costo di trasferimento ab_i;
  analogamente, se lo stadio di lavorazione i avviene sulla linea B e lo stadio di lavorazione $i+1$ avviene sulla linea A, si deve aggiungere il costo di trasferimento ba_i.
- Per ogni $i = 0, \ldots, n-2$, se gli stadi di lavorazione i e $i+1$ avvengono sulla stessa linea, allora non si paga alcun costo di trasferimento.

Formalmente, sia $X = \langle x_0, \ldots, x_{n-1}\rangle$ una n-pla tale che $x_i = A$ o $x_i = B$ per ogni $i = 0, \ldots, n-1$, Allora il costo totale $C(X)$ di X è dato da

$$\sum_{i=0}^{n-1} c_i + \sum_{i=0}^{n-2} t_i$$

dove, per ogni $i = 0, \ldots, n-2$:

$$t_i = \begin{cases} ab_i & \text{se } x_i = a_i \text{ e } x_{i+1} = b_i \\ ba_i & \text{se } x_i = b_i \text{ e } x_{i+1} = a_i \\ 0 & \text{altrimenti} \end{cases}$$

Anche in questo caso, lo scopo dell'esercizio è calcolare una n-pla X tale che il Il costo $C(X)$ sia minimo tra tutte le possibili scelte.

I costi di lavorazione delle due catene e i relativi costi di trasferimenti sono contenuti in un file di dati nel formato seguente:

n
a_0 a_1 ... a_{n-1}
b_0 b_1 ... b_{n-1}
ab_0 ab_1 ... ab_{n-2}
ba_0 ba_1 ... ba_{n-2}

Il programma deve:

1. Chiedere all'utente il nome del file dati (un file `.txt`) da cui leggere i costi di lavorazione e di trasferimento.
2. Leggere i dati dal file specificato e caricare i costi di lavorazione in due array di n elementi ciascuno (oppure in un unico array di $2n$ elementi), e i costi di trasferimento in due ulteriori array di $n-1$ elementi (oppure un ulteriore array di $2(n-1)$ elementi) allocati dinamicamente. (Si suppone che il formato del file sia sempre corretto).
3. Calcolare una n-pla $\langle x_0, \ldots, x_{n-1} \rangle$ di costo minimo.
4. Stampare la soluzione associata a tale n-pla nel formato esemplificato qui di seguito.

Esempio Si supponga che il file di input sia:

```
7
2 7 9 1 4 3 1
5 1 4 3 7 8 2
4 3 2 4 3 5
1 6 8 5 4 1
```

Allora l'output va visualizzato come segue:

```
Costo totale: 26

--  --  --  --   4   3   1
              5
 5   1   4   3  --  --  --
```

Suggerimenti

Per il calcolo della n-pla X di costo minimo si consiglia di utilizzare il paradigma della programmazione dinamica: in estrema sintesi, soluzione dei sottoproblemi e memorizzazione delle soluzioni.

In particolare, ci si doti di quattro array ausiliari:
`int *sumA, *sumB`: allocati dinamicamente, di dimensione n.
`char *predA, *predB`: allocati dinamicamente, di dimensione $n-1$;

Il generico elemento `sumA[i]` conterrà il costo di una $(n-i)$-pla $\langle x_i, x_{i+1}, \ldots, x_{n-1} \rangle$ di costo minimo tra tutte le $(n-i)$-ple tali che $x_i = A$. Analogamente l'elemento `sumB[i]` conterrà il costo di una $(n-i)$-pla $\langle x_i, x_{i+1}, \ldots, x_{n-1} \rangle$ di costo minimo tra tutte le $(n-i)$-ple tali che $x_i = B$.

La costruzione degli array `sumA` e `sumB` avviene scandendoli da destra a sinistra, vale a dire:

1. all'inizio si pone `sumA`$[\mathtt{n}-1] = a_{n-1}$, `sumB`$[\mathtt{n}-1] = b_{n-1}$.

2. al generico passo i, il valore di `sumA[i]` è ottenuto in base ai valori di `sumA[i+1]` e `sumB[i+1]` e contemplando l'eventuale costo di trasferimento ab_i.
   Analogamente si determina il valore di `sumB[i]`.

3. Alla fine in `sumA[0]` e `sumB[0]` troviamo i costi di due soluzioni X_A e X_B, con il primo elemento di X_A appartenente alla linea A e il primo elemento di X_B appartenente alla linea B. A questo punto basta considerare la soluzione di costo minimo fra le due.

Abbiamo così ottenuto il costo di una soluzione ottimale, ma per determinare effettivamente la struttura della soluzione, usiamo gli array `predA`, `predB`.

Il generico elemento `predA[i]` varrà 0 se nella sottosoluzione $\langle x_i, x_{i+1}, \ldots, x_{n-1}\rangle$ con $x_i = A$, l'elemento x_{i+1} appartiene alla linea A, `predA[i]` varrà 1 se invece l'elemento x_{i+1} appartiene alla linea B.
`predB[i]` varrà 0 se nella sottosoluzione $\langle x_i, x_{i+1}, \ldots, x_{n-1}\rangle$ con $x_i = B$, l'elemento x_{i+1} appartiene alla linea B, `predB[i]` varrà 1 se invece l'elemento x_{i+1} appartiene alla linea A.

La costruzione degli array `predA`, `predB` avviene contestualmente alla costruzione degli array `sumA` e `sumB`: nel momento in cui costruiamo `sumA[i]` abbiamo tutte le informazioni per determinare `predA[i]`.

Quando la costruzione dei quattro array ausiliari è stata completata, possiamo usarli per ricostruire una soluzione ottimale.

1. Si determina il primo elemento confrontando i valori di `sumA[0]` e `sumB[0]`.

2. Si determina l'$(i+1)$-elemento considerando l'i-esimo elemento e il valore di `predA[i]` o `predB[i]`.

Nota: anche in questo caso, può essere utile ridurre il numero di array ausiliari a due: `sum` che è pensato come una tabella $2 * n$ e `pred` che è pensato come una tabella $2 * (n-1)$. Chi opta per questa implementazione potrà trovare utile definire una macro:
`#define posto(n,col,row) ((n)*(col)+(row))`
per accedere agli elementi dell'array monodimensionale pensato come tabella bidimensionale.

Esempio
Il file contenga i seguenti dati:

```
3
6 5 1
3 4 7
6 8
2 4
```

Allora i quattro array ausiliari vengono costruiti in 3 passi come segue:

1. Passo 0:

   ```
   sumA:  ?  ?  1   predA: ? ?
   sumB:  ?  ?  7   predB: ? ?
   ```

2. Passo 1:

   ```
   sumA:  ?  6  1   predA: ? 0
   sumB:  ?  9  7   predB: ? 1
   ```

3. Passo 2:

```
sumA: 12  6  1   predA: 0 0
sumB: 11  9  7   predB: 1 1
```

E dunque la soluzione è codificata dalla 3-tupla $X = \langle B, A, A \rangle$, dato che: $11 < 12$, `predB[0] = 1` e `predA[1] = 0`. Il programma deve quindi stampare:

```
Costo totale: 11

--   5   1
   2
 3  --  --
```