Fondamenti di Informatica, A.A. 2011-2012

11/07/2012

Prova Pratica

Si deve gestire la situazione dei biglietti di un teatro. I posti possono essere rappresentati con una mappa con valore 0 per posto libero, 1 per posto occupato; il costo del singolo biglietto è dato, uniforme lungo ciascuna fila, e variabile da fila a fila. Tutte le file hanno lo stesso numero di posti.

Ciascun potenziale cliente richiede un certo numero di posti liberi da assegnare tutti sulla stessa fila (anche se non necessariamente contigui), indicando anche il massimo prezzo che è disposto a pagare.

Scrivere una funzione Matlab che, nel caso in cui sia possibile soddisfare la richiesta, restituisca la fila a cui si trovano i posti, il costo effettivamente pagato e la mappa dei posti aggiornata con i posti appena assegnati; la funzione dovrebbe cercare di massimizzare il profitto del teatro, quindi vendere al prezzo più alto compatibile con i vincoli del cliente.

Opzionale: se non esiste nessuna fila che rispetta i vincoli, tentare di risolvere il problema con la fila, se esiste, che abbia abbastanza posti liberi; in questo caso, si deve proporre il costo risultante al cliente senza aggiornare immediatamente la mappa.

Svolgimento

Il problema dato può essere affrontato assumendo di avere nel workspace due dati persistenti per una sessione di prenotazioni, ossia la mappa dei posti ed un vettore che per ogni riga dia il costo del singolo biglietto.

Ogni evento di tentata prenotazione sarà simulato dalla esecuzione di una funzione che tenta di soddisfare la richiesta, secondo il testo, e restituisce la mappa aggiornata, che sarà poi usata come input della prenotazione successiva; il vettore dei costi non è soggetto a cambiamenti da parte della funzione. La funzione Matlab che realizza la soluzione viene riportata di seguito:

```
function [cost,row,map,newcost] = findplace(num,maxcost,map,rcost)
  %
  % Find a row in MAP with NUM free seats.
  % Each place in row I has a cost RCOST(I);
  % the customer wants NUM places at maximum MAXCOST.
  % The subroutine should find the ROW with NUM free seats
  % that maximizes the profit within the user constraints;
  % the actual price to be paid is COST.
  % MAP is updated with the new occupation.
  % If no suitable row exists, return the minimum cost
  % NEWCOST at which there are enough free seats,
  % but MAP is not updated.
  %

  newcost=0;
```

```
% 1. Figure out where the free seats  are.
frees = size(map,2)-sum(map,2);
if (num>max(frees))
  disp('Not enough free seats in any row!');
  return
end
% 2. Figure out the cost of NUM tickets in each row
costs = num .* rcost;
% 3. Cut away rows without enough free seats
costs((frees < num)) = 0;
% 4. Cut away rows that cost too much
costs((costs >maxcost)) = 0;
% 5. Those left are all candidate solutions.
%    Choose the one maximizing profits.
[cost,row]=max(costs);
if (cost>0)
  % 6. There is at least one suitable row; update map.
  k=0;
  j=1;
  while (j<=size(map,2)) && (k<num)
    if (map(row,j)==0)
      map(row,j)=1;
      k=k+1;
    end
    j=j+1;
  end
else
  % 7. No suitable row exists. Try the cheapest row
  %    with enough free seats, if any.
  costs = num .* rcost;
  huge=max(costs)*10;
  costs((frees < num)) = huge;
  [newcost row]=min(costs);
end
```