Soluzione del I compitino di programmazione dell' 11 febbraio 2009

Si richiede di trattare un array T a 1 dimensione di dim elementi come un array a due dimensioni con colon colonne. Quindi questo array avrà R= dim/colon righe piene ed eventualmente un'ultima riga con solo E=dim%colon elementi. Sulle righe di questa matrice (anche l'ultima con E elementi, se E>0) va fatto un pattern match di un array P. E' richiesto di contare i match di P su ciascuna riga, ma considerando solo match non sovrapposti tra di loro. La tecnica per affrontare il problema è quella di dividerlo in sotto problemi ognuno dei quali verrà risolto con una funzione. I sottoproblemi sono:

1) fare il match di P su una riga dato un punto d'inizio. La funzione ha bisogno di: P, dim_P, e puntatore all'elemento della riga da cui iniziare il match. La funzione restituisce un booleano per dire se il match è riuscito o no. Necessario garantire che dal punto d'inizio del match, sicuramente la riga contenga abbastanza elementi per completare il match, cioè gli elementi siano almeno dim_P Questo evita di dover controllare che si resta all'interno della riga.
2) Cercare e contare tutti i match non sovrapposti in una riga.
3) Esaminare tutte le righe di T (anche l'ultima, possibilmente più corta), per ciascuna calcolare quanti match non sovrapposti ci sono per quella riga, ricordando l'indice della riga che contiene il massimo numero di match non sovrapposti. In caso più righe abbiano il massimo numero di match, si restituisce l'indice di una qualsiasi delle righe, per esempio il più piccolo.

Funzione del punto (1):

```
bool P1(char *T, char *P, int dim_P)
{
bool ok= true;
for(int i=0; i<dim_P && ok; i++) // 0<=i<=dim_P, ok⇔ T[0..i-1]==P[0..i-1]
        if(T[i]!=P[i])
                ok=false;
return ok;
}// restituisce true sse T[0..dim_P-1]=P[0..dim_P-1]
```

Funzione del punto (2):

```
int P2(char* T, int dim, char* P, int dim_P)
{
int i=0, conta=0,skip=1;
for(  ;i<dim-dim_P+1;i=i+skip) // conta è n. match di P non sovrapposti  con inizio in T[0..i-skip]
// skip=dim_P => T[i-dim_P] è inizio match
//skip=1 => T[i-1] inizio mismatch
{
        if(P1(T+i,P,dim_P))
        {
                skip=dim_P; // evita match sovrapposti
                conta++;
        }
        else
                skip=1;

}
// conta è n. match di P non sovrapposti  con inzio in T[0..dim-dim_P] per avere la post-condizione basta
// osservare che in T[dim-dim_P+1..dim-1] nessun match completo può iniziare
return conta;
}// restituisce numero di match non sovrapposti di P in T[0..dim-1]
```

La funzione del punto (3):

```
int P3(char* T, int dim, int colon, char* P, int dim_P)
{
int righec=dim/colon, eleu=dim%colon, maxm=0, pmaxm=0;

// T vista come array R * colon ha righec righe di colon elementi e possibilmente un'ultima riga con solo
// eleu elementi. Quindi se eleu>0, R=righec+1  e altrimenti R=righec.

for(int i=0; i<righec; i++) // maxm e pmaxm individuano la riga in 0..i-1 con max n. di match di P non
//sovrapposti
        {
                int k=P2(T+i*colon,colon,P,dim_P);
                if(k>maxm)
                {
                maxm=k;  pmaxm=i;
                }

        }
if(eleu>=dim_P) // c'è qualcosa nell'ultima riga ?
        {
        int k=P2(T+righec*colon, eleu,P,dim_P);
        if(k> maxm)
        pmaxm=righec;
        } // pmaxm riga con max n. di match di P non  sovrapposti  considerando anche l'ultima riga
return pmaxm;
}// restituisce
```