Combinació d'iteració i recursivitat

R. Ferrer i Cancho

Universitat Politècnica de Catalunya

PRO2 (curs 2010-2011)
Versió 0.0

Avís: aquesta presentació no pretén ser un substitut dels apunts oficials de l'assignatura.

On som?

- Tema 8: Combinació de recursivitat i iteració
- 12a sessió

Avui

- Problemes de cerca exhaustiva
- Problemes combinatoris: subconjunts, permutacions,...

El problema de la motxilla (versió simplificada)

- "Pesos": $p_1, ..., p_i, ..., p_N$
- Capacitat k
- Minimitzar

$$k - \sum_{i=1}^{N} b_i p_i \geq 0 \quad (1)$$

- Amb $b_i \in \{0, 1\}$
- No superar k.

```
int pes(const vector<int> &p, int k)
{
// Pre: cert

// Post: retorna el "mínim sobrant resp. k amb elements de
//       p màxim un cop i no superant k"
}
```

Funció d'immersió

```
int i_pes(const vector<int> &p, int i, int k)
{
// Pre: i< p.size(); k>= 0
  int s;

  if (i<0) s=k;
  else if (k==0) s=0;
  else {
    s= i_pes(p,i-1,k);
       // H.I.: retorna el "mínim sobrant resp. k amb elements de p[0..i-1]
       //       màxim un cop i no superant k"
    if (p[i]<=k) {
    int s1= i_pes(p,i-1,k-p[i]);
       // H.I.: retorna el "mínim sobrant resp. k-p[i] amb elements de p[0..i-1]
       //       màxim un cop i no superant k-p[i]"
    if (s1<s) s=s1;
    }
  }
  return s;
// Post: retorna el "mínim sobrant resp. k amb elements de p[0..i]
//       màxim un cop i no superant k"
}
```

Crida a la funció d'immersió

```
int pes(const vector<int> &p, int k)
{
// Pre: cert
  return i_pes(p,p.size()-1,k);
// Post: retorna el "mínim sobrant resp. k amb elems de p
//       màxim un cop i no superant k"
}
```

Variant: retornar una solució

Solució:

- sol: vector amb els indexos dels pesos triats.
- l: nombre de cel·les ocupades de sol.

```
int pessol(const vector<int> &p, int k, vector<int> &sol, int &l)
{
// Prec: p.size() = sol.size()
  ...
// Post: retorna el "mínim sobrant respecte k amb elements de p
//       màxim un cop i no superant k";
//       sol conté la solució trobada amb llargada l
}
```

Funció d'immersió

```
int i_pessol(const vector<int> &p, int i, const int k,vector<int> & sol, int &l)
{
// Pre: i< p.size(); p.size() = sol.size()
  int s;

  if (i<0) {s=k;l=0;}
  else if (k==0) {s=0; l=0;}
  else {
    s= i_pessol(p,i-1,k,sol,l);
    if (p[i]<=k) {
      int l1;
      vector<int> sol1(p.size());
      int s1= i_pessol(p,i-1,k-p[i],sol1,l1);
      if (s1<s) {s=s1; sol1[l1]=p[i]; sol = sol1; l=l1+1;}
    }
  }
  return s;
// Post: retorna el "mínim sobrant respecte k amb elements de
//       p[0..i] màxim un cop i no superant k";
//       sol conté la solució trobada amb llargada l
}
```

Crida a la funció d'immersió

```
int pessol(const vector<int> &p, int k, vector<int> &sol, int &l)
{
// Prec: p.size() = sol.size()

return i_pessol(p,p.size()-1,k,sol,l);
// Post: retorna el "mínim sobrant respecte k amb elements de p
//       màxim un cop i no superant k";
//       sol conté la solució trobada amb llargada l
}
```

Motxilla amb matriu de pesos

- Matriu de pesos d'objectes (prestatgeria).
- De cada fila (prestatge) només podem emprar-ne un.

```
typedef vector<vector<int> > matriu;

int mpes(const matriu &m, const int k)
{
// Pre: cert
...
// Post: retorna el "mínim sobrant resp. k amb elems de les files de m
//       max. un per fila i no superant k"
}
```

Funció d'immersió

```
int i_mpes(const matriu &m, int i, const int k)
{
// Pre: i< p.size()
  int s;

  if (i<0) s=k;
  else if (k==0) s=0;
  else {
    s = i_mpes(m,i-1,k);
    for (int j=0; j<m[i].size();++j){
      if (m[i][j]<=k) {
        int s1= i_mpes(m,i-1,k-m[i][j]);
        if (s1<s) s=s1;
      }
    }
  }
  return s;
// Post: retorna el "mínim sobrant resp. k amb elems de les files de m[0..i]
//       max. un per fila i no superant k"
}
```

Crida a la funció d'immersió

```
int mpes(const matriu &m, const int k)
{
// Pre: cert

  return i_mpes(m,m.size()-1, k);

// Post: retorna el "mínim sobrant resp. k amb elems de les files de m
//       max. un per fila i no superant k"
}
```

Mostrar tots els subconjunts possibles

Especificació:

```
void subset(const vector<int> &p)
{
// Pre: cert
...
// Post: mostra tots els subconjunts de p en línies diferents seguint,
//       per a cada línia, el format
//       mida del conjunt ":" elements del subconjunt.
}
```

El conjunt buit és un dels subconjunts.

Acció d'immersió

```
void i_subset(const vector<int> &p, int i, vector<int> &sol, int &l)
{
// Pre: 0=<i=<p.size(); sol[0..l-1] es un subconjunt de p[0..i-1]
  if (i==p.size()) {cout << l << ": "; escriu_vector(sol,l); }
  else {
    int l1 = l;
    i_subset(p,i+1,sol,l); // HI: mostra tots els conjunts ... sense p[i] ...
    sol[l1]=p[i];
    ++l1;
    i_subset(p,i+1,sol,l1); // HI: mostra tots els conjunts ... amb p[i] ...
  }
// Post: mostra tots els conjunts que comencen pel subconjunt sol[0..l-1] i
//       acaben per un subconjunt de p[i..p.size()-1] en línies diferents
//       seguint, per a cada línia, el format
//       mida del conjunt ":" elements del subconjunt
}
```

- Mostra un mateix subconjunt més d'un cop?
- Es deixa algun subconjunt?

Acció auxiliar d'escriptura

```
void escriu_vector(const vector<int>& v, int l)
{
  for (int i=0; i<l;++i)
    {
      cout << v[i] << " " ;
    }
  cout << endl;
}
```

Crida a la funció d'immersió

```
void subset(const vector<int> &p)
{
// Pre: cert
vector<int> sol(p.size());
int l=0;
i_subset(p,0,sol,l);
// Post: mostra tots els subconjunts de p en línies diferents seguint,
//       per a cada línia, el format
//       mida del conjunt ":" elements del subconjunt.
}
```

Mostrar totes les permutacions possibles

Especificació:

```
void permut(const vector<int> &p)
{
// Pre: cert

// Post: mostra totes les permutacions dels elements de p
}
```

Fer servir acció auxiliar:

```
void escriu_vector(const vector<int>& v)
{
  for (int i = 0; i < v.size(); ++i)
    {
      cout << v[i] << " " ;
    }
  cout << endl;
}
```

Acció d'immersió

```
void i_permut(const vector<int> &p, vector<bool> &sel, vector<int> &sol, int l)
{
// Pre: 0=<l=<p.size(); sol[0..l-1] es una permutació dels valors de p
//      en posicions seleccionades a sel; l=nombre de seleccionats de sel
  if (l==p.size()) { escriu_vector(sol); }
  else {
    int j=0;
    // Inv: hem mostrat les permutacions que comencen per sol[0..l],
    //      on sol[l] pren valors de p[0..j-1] no seleccionats, i acaben
    //      per una permutacio dels no seleccionats
    while (j < sel.size()) {
      if (not sel[j]){
        sel[j]=true;
        sol[l]=p[j];
        i_permut(p, sel, sol, l+1); // HI: mostra totes les permut ... amb p[j] a sol[l]
        sel[j]=false;
      }
      ++j;
    }
  }
// Post: mostra totes les permutacions que comencen per sol[0..l-1] i
//       acaben per una permutacio p de les posicions no seleccionades en sel
}
```

Crida a l'acció d'immersió

```
void permut(const vector<int> &p)
{
// Pre: cert
vector<int> sol(p.size());
vector<bool> sel(p.size(),false);

i_permut(p, sel, sol, 0);
// Post: mostra totes les permutacions dels elements de p
}
```