INF280
Stratégies de recherche

Antoine Amarilli

23 juin 2014

Introduction

- Structures générales pour résoudre des problèmes
- Se demander : cette méthode est-elle applicable ici ?
- Ne couvre pas tout !

Table des matières

Principe

- Algorithme glouton : faire le choix localement meilleur
- Ne jamais revenir sur ses choix
- (Anglais : greedy algorithm)

Exemple de glouton : reconnaître un graphe biparti

Entrée
- graphe orienté G
- racine $root$

Hypothèse Tous les sommets de G sont accessibles depuis $root$

Sortie Déterminer si G est biparti ?

Exemple de glouton : reconnaître un graphe biparti

Entrée
- graphe orienté G
- racine $root$

Hypothèse Tous les sommets de G sont accessibles depuis $root$

Sortie Déterminer si G est biparti ?

$\Rightarrow$ **Intuition :** si G est biparti, alors bipartition unique
- ... excepté la symétrie entre les parties 1 et 2

Exemple de glouton : code

```
int color(int v, int c) {
  if (col[v])
    return col[v] == c ? 1 : 0;
  col[v] = c;
  for (unsigned int i = 0; i < adj[v].size(); i++)
    if (!color(adj[v][i], -c))
      return 0;
  return 1;
}

for (int i = 0; i < N; i++)
  col[i] = 0;
color(root, 1);
```

Gloutons et tris

- Importance de l'ordre des choix
- Parfois, glouton seulement possible pour le bon ordre
- Souvent, il faut trier pour avoir le bon ordre
- Se demander : le glouton suivant tel tri est-il optimal ?

Exemple de glouton et tri : choix d'activités

Entrée Activités avec date de début et de fin $d_i < f_i$

Sortie Sous-ensemble maximal sans chevauchement

Exemple de glouton et tri : choix d'activités

Entrée Activités avec date de début et de fin $d_i < f_i$

Sortie Sous-ensemble maximal sans chevauchement

$\Rightarrow$ **Intuition :** trier les activités par date de fin croissante

Exemple de glouton et tri : code

```
for (int i = 0; i < N; i++) {
  scanf("%d %d", &d, &f);
  v.push_back(make_pair(f, d));
}
sort(v.begin(), v.end());
int nok = 0, last = -1;
for (int i = 0; i < N; i++) {
  if (v[i].second < last)
    continue;
  nok++;
  last = v[i].first;
}
```

Exemple de glouton et tri : justification

- Considérons une solution optimale
- Considérons le tri par date de fin croissante
- Considérons la première activité *a* que l'optimale ne prend pas
- On peut remplacer l'activité suivante de l'optimale par *a*
- On obtient une solution
  - aussi bonne que l'optimale
  - qui fait le choix glouton
- Induction, on répète le processus

$\Rightarrow$ La solution gloutonne est une solution optimale

Table des matières

Principe

- Dichotomie :
  - $\Rightarrow$ recherche d'un élément dans un tableau trié
- Généralisation :
  - $\Rightarrow$ recherche d'une frontière entre deux régions
- Coût seulement logarithmique (contre-intuitif !)

Exemple de dichotomie : couverture de points

Entrée.
- points P sur un segment
- nombre K de points

Sortie. K points P' telle que la distance maximale d'un point de P à un point de P' soit minimale.

Exemple de dichotomie : couverture de points

Entrée.
- points P sur un segment
- nombre K de points

Sortie. K points P' telle que la distance maximale d'un point de P à un point de P' soit minimale.

$\Rightarrow$ Idée : dichotomiser sur la distance maximale

Exemple de dichotomie : suite

- Une distance maximale est soit réalisable soit irréalisable
- Monotonie :
  - si d est réalisable alors tout $d' < d$ l'est aussi
  - si d est irréalisable alors tout $d' > d$ l'est aussi

  $\Rightarrow$ une seule frontière entre réalisable et irréalisable
- Pour une distance maximale D fixée, algorithme glouton

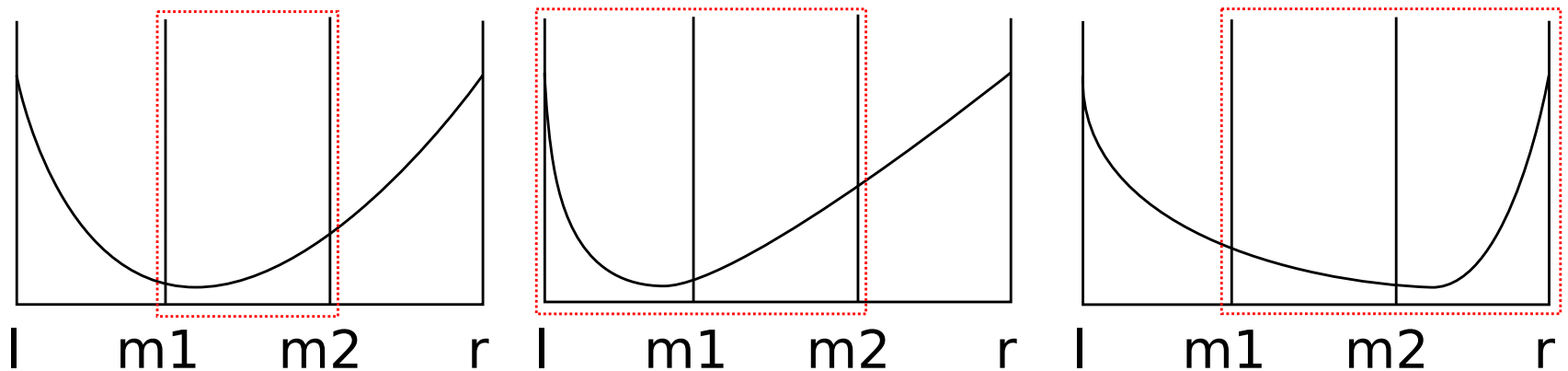
$\Rightarrow$ Dichotomie puis glouton

Trichotomie (ternary search)

- Dichotomie : recherche d'une valeur cible
- Que faire si la cible est un optimum inconnu ?
- Hypothèse : la fonction a un seul optimum local

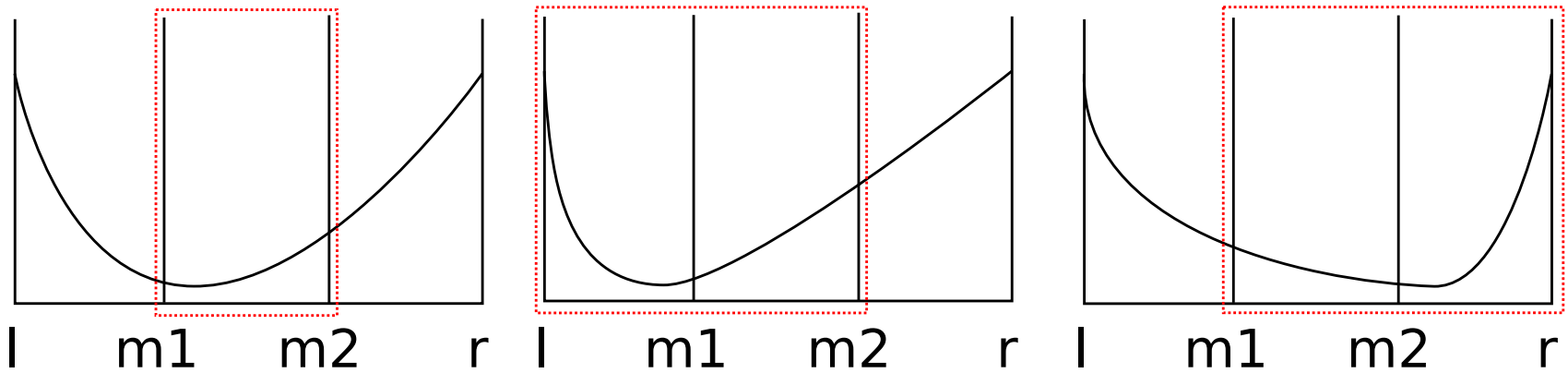
Trichotomie (ternary search)

- Dichotomie : recherche d'une valeur cible
- Que faire si la cible est un optimum inconnu ?
- Hypothèse : la fonction a un seul optimum local



Trichotomie (ternary search)

- Dichotomie : recherche d'une valeur cible
- Que faire si la cible est un optimum inconnu ?
- Hypothèse : la fonction a un seul optimum local



$\Rightarrow$ Recherche ternaire

Trichotomie : exemple

Entrée
- points P du plan
- segment s

Sortie Point de s qui minimise la distance max aux points P.

Trichotomie : exemple

Entrée
- points P du plan
- segment s

Sortie Point de s qui minimise la distance max aux points P.

$\Rightarrow$ Idée : le min du max a un unique minimum sur le segment...

$\Rightarrow$ (Aussi possible de faire une dichotomie sur la réponse ici.)

Trichotomie : code

```
double f(double x) {
  // ... calculer la distance max des points de P
  // ... au point x sur le segment s
}

double tricho(double l, double r) {
  if (r - l < EPS)
    return l;
  double m1 = (l + r + r) / 3, m2 = (l + l + r) / 3;
  if (f(m1) > f(m2))
    return tricho(m1, r);
  else
    return tricho(l, m2);
}

printf("%lf\n", tricho(0, MAX))
```

Table des matières

Principe

- Ordre sur les sous-problèmes
- Dynamique : résoudre des plus petits aux plus grands cas
- Récursif mémoisé : des plus grands aux plus petits
- Choix à chaque niveau : coût local + coût du sous-problème

$\Rightarrow$ Déjà traité aux séances précédentes.

Table des matières

Principe

- Ordre sur les choix
- Énumérer les options possibles
  - Pour chacune, la faire et tenter de continuer
- Si coincé, revenir en arrière (backtrack)
- Souvent possible de faire un choix en place

Exemple : sudoku

```
int g[9][9];
int solve(int i) {
  int x = i/9, y = i%9;
  if (x >= 9) return 1;
  if (g[x][y] != 0)
    return solve(i+1);
  for (int k = 1; k <= 9; k++)
    if (acceptable(x, y, k)) {
      g[x][y] = k;
      if (solve(i+1))
        return 1;
      g[x][y] = 0;
    }
  return 0;
}
```

Cas particulier : minimax

- Backtrack usuel : si une branche réussit alors c'est réussi.
- Cas du jeu à deux joueurs :
  - Une configuration est gagnée par le joueur au trait...
  - ... si l'une des accessibles est gagnée par lui.
  - Inversement, si toutes les accessibles sont perdues pour lui...
  - ... alors la configuration courante est perdante pour lui.

Exemple : minimax

```
int minimax(int i, int p) {
  int winner;
  if (winner = game_over())
    return winner;
  for (int m = 0; m < nmoves; m++)
    if (admissible(m, p)) {
      do_move(m, p);
      int ret = minimax(i+1, -p);
      undo_move(m, p);
      if (ret == p)
        return p;
    }
  return -p;
}
```

Heuristiques

- Ordre des choix :
  - Privilégier les choix certains
  - Privilégier les choix avec peu d'options
- Ordre des options :
  - Privilégier les options contraignantes

$\Rightarrow$ Glouton : cas idéal, jamais de backtrack pour cet ordre

Pruning

- Booléen : tout terminer dès qu'une solution est trouvée
- Numérique : couper les branches pires que l'optimum courant
- Propagation de contraintes :
  - observer les conséquences de l'option retenue
  - voir si on ne s'est pas coincé pour plus tard
- Pour minimax : alpha-beta pruning

Table des matières

Principe

- Énumérer toutes les solutions possibles
- Vérifier en bloc si elles sont bonnes
- Attention à la performance !
  - Nombre de solutions
  - Coût de la vérification

  $\Rightarrow$ Vérification non incrémentale

  $\Rightarrow$ Solutions partielles invalides non rejetées

Énumérer les permutations

```
int p[MAXN];
for (int i = 0; i < N; i++)
  p[i] = i;
do {
  // tester la permutation p
  // ...
} while (next_permutation(p, p+N));
```

Énumérer les sous-ensembles (N petit)

```
for (int s = 0; s < (1 << N); s++) {
  // s en binaire est un sous-ensemble de {0, ..., N-1}
  // (s & (1 << i)) pour savoir si i est dans l'ensemble
  // voir slides sur la manipulation de bits
  // ...
}
```

Table des matières

Conclusion

- Gardez à l'esprit ces schémas classiques
- Bien sûr, combinaisons :
  - ⇒ Bruteforce sur un choix puis glouton sur les autres
  - ⇒ Dichotomie sur un paramètre puis glouton avec le bon tri
  - ⇒ etc.