STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 21

Choisir localement et justifier globalement

Sélectionnez un maximum d'activités compatibles par date de fin. Justifiez le choix glouton et reconnaissez un cas où un autre choix local échoue.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Justifier un choix local par un argument d'échange.
- Un choix glouton nécessite une preuve adaptée à l'objectif.
- Pour maximiser le nombre d'intervalles compatibles, choisir la fin la plus précoce admet un argument d'échange.

Prérequis. Avoir étudié les chapitres DSA précédents ; connaître recherche, tris et raisonnement sur les coûts.

Partir d'un besoin

Une salle accueille des activités sans chevauchement. Choisir la plus courte paraît raisonnable, mais maximise-t-il toujours le nombre d'activités ? Il faut justifier la règle de choix.

Définir la structure et les conventions

Une méthode **gloutonne** effectue un choix local qu'elle ne remet plus en cause. Sa correction demande une propriété du problème ; l'intuition seule ne suffit pas. Nous cherchons le plus grand **nombre** d'activités compatibles, sans poids ni valeur différente entre activités.

Une activité est un intervalle [début, fin) : la fin est exclue. Une activité qui commence à la fin de la précédente est donc compatible. Les intervalles ont des bornes finies et début < fin. La règle est : trier par fin croissante, puis accepter une activité si son début est au moins égal à la fin de

la dernière acceptée.

L'**argument d'échange** justifie ce choix : dans une solution optimale, remplacer la première activité par celle qui finit le plus tôt ne bloque aucune des suivantes. Il reste donc une solution de même taille commençant par notre choix. On répète le raisonnement sur les activités encore compatibles.

Python sorted(..., key=lambda interval: interval[1]) trie selon le deuxième champ ; lambda décrit ici cette petite fonction de clé. Les couples contiennent début puis fin. MATLAB place les intervalles en lignes et utilise sortrows(..., 2) pour trier selon la deuxième colonne.

Suivre les états

Choix par fin croissante

Activité	Dernière fin avant le choix	Décision
[0, 3)	Aucune	Accepter
[1, 4)	3	Refuser
[3, 5)	3	Accepter
[5, 7)	5	Accepter

Comparer les réalisations

Trois activités sont retenues : [0, 3), [3, 5) et [5, 7). MATLAB utilise : pour prendre toute une ligne. Le tableau selected est réservé à la taille maximale, puis réduit aux lignes utilisées. Le résultat ne change pas les intervalles de départ.

Pseudo-code

```
// example.pseudo
// select_activities
// Sort by increasing end time.
// Keep an activity when start >= end of last selected activity.
// The objective is the number of activities, not total duration or value.
```

Python

```python
# select_activities.py
def select_activities(intervals):
    ordered = sorted(intervals, key=lambda interval: interval[1])
    selected = []
    last_end = float("-inf")
    for start, end in ordered:
        if start >= last_end:
            selected.append((start, end))
            last_end = end
    return selected
intervals = [(0, 3), (1, 4), (3, 5), (5, 7)]
selected = select_activities(intervals)
print(selected)
```

MATLAB

```
% select_activities_demo.m
intervals = [0, 3; 1, 4; 3, 5; 5, 7];
selected = select_activities(intervals);
disp(selected);
function selected = select_activities(intervals)
    ordered = sortrows(intervals, 2);
    selected = zeros(size(intervals));
    count = 0;
    last_end = -inf;
    for row = 1:size(ordered, 1)
        if ordered(row, 1) >= last_end
            count = count + 1;
            selected(count, :) = ordered(row, :);
            last_end = ordered(row, 2);
        end
    end
    selected = selected(1:count, :);
end
```

Justifier et compter

Après un tri en O(n log n), le parcours de sélection est Θ(n). La sortie occupe O(n) mémoire ; le tri a ses propres besoins de stockage. Les fonctions de tri standards sont utilisées pour cette étape, sans reconstituer leur implantation interne. Le choix par date de fin ne résout pas automatiquement la sélection d’activités pondérées.

Prévoir puis expérimenter

Essayez les intervalles [0, 3), [3, 6) et [2, 4). Choisir la durée la plus courte prendrait [2, 4) et bloquerait les deux autres ; choisir la fin la plus précoce en conserve deux. Cela sépare une intuition locale d’une règle prouvée.

Exercice corrigé

▼ Pourquoi ne peut-on pas garder cette preuve si chaque activité possède un gain différent ?

Un échange peut conserver le nombre d'activités tout en diminuant leur gain total. Le nouvel objectif exige une autre justification et souvent une programmation dynamique.

À retenir

- Un choix glouton nécessite une preuve adaptée à l'objectif.
- Pour maximiser le nombre d'intervalles compatibles, choisir la fin la plus précoce admet un argument d'échange.
- Changer l'objectif en gain total invalide cette preuve.

Version web : https://www.clearcode.fr/courses/programming/dsa/strategies/greedy