Максимізація прибутку дропшипінгу

Кльоц Богдан, КМ-01

ЗАДАЧА ПРОЕКТУ

Є ПЕВНА КІЛЬКІСТЬ ЗАМОВЛЕНЬ З НАСТУПНИМИ ХАРАКТЕРИСТИКАМИ: ВАРТІСТЬ ЗАМОВЛЕННЯ, ПРИБУТКОВІСТЬ, ЧАС НЕОБХІДНИЙ НА ВИКОНАННЯ. НЕОБХІДНО ОТРИМАТИ МАКСИМАЛЬНИЙ ПРИБУТОК ЗА НАЯВНОГО КАПІТАЛУ І ЧАСУ. НАПРИКЛАД, 30 ЗАМОВЛЕНЬ, 30 ДНІВ, 3000$.

ТЕОРІЯ

Задача часто виникає при розподілі ресурсів, коли наявні фінансові обмеження, і вивчається в таких областях, як комбінаторика, інформатика, теорія складності, криптографія, прикладна математика.

Задача комбінаторної оптимізації: для заданої множини предметів, кожен з яких має вагу і цінність, визначити яку кількість кожного з предметів слід взяти, так, щоб сумарна вага не перевищувала задану, а сумарна цінність була максимальною.

Описання задачі досить просте, але розв'язання — складне. Відомі алгоритми, на практиці, здатні розв'язати задачі досить великих розмірів. Однак, унікальне формулювання задачі, а також той факт, що вона присутня як підзадача у більших, загальніших проблемах, робить її важливою для наукових досліджень.

Методи розв’язання

ЖАДІБНИЙ АЛГОРИТМ

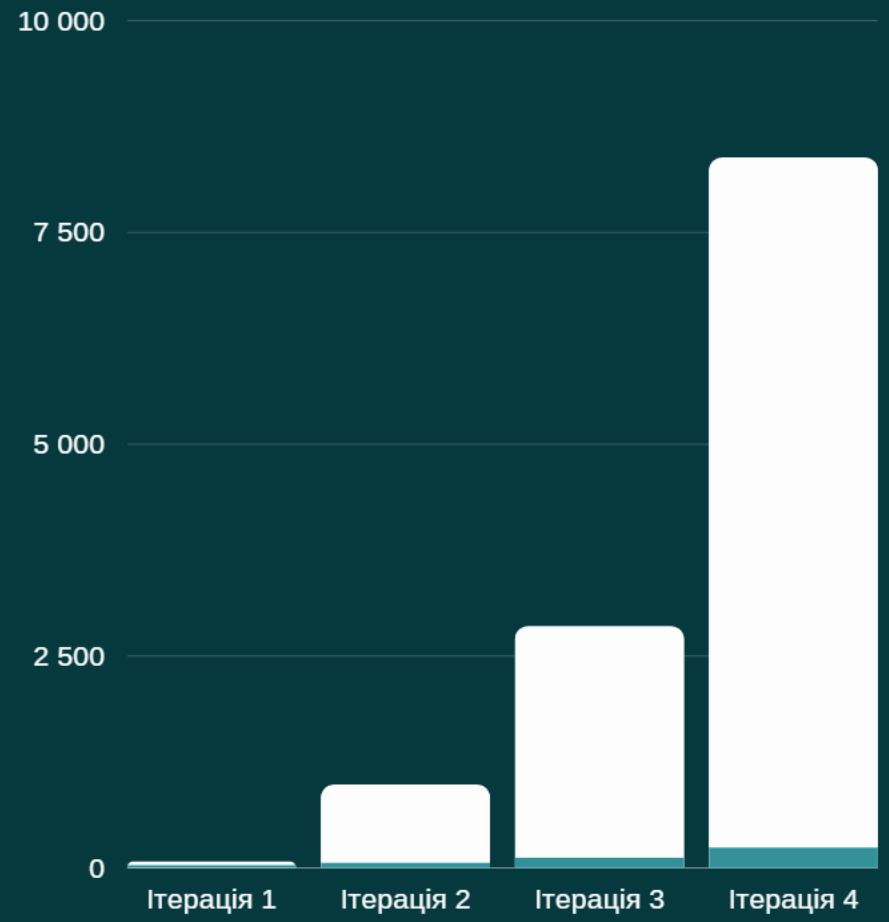
Один з найпростіших алгоритмів пошуку приблизного розв'язку. Ідея алгоритму полягає в тому, щоб брати найприбутковіші замовлення для яких потрібен мінімум часу.

ДИНАМІЧНЕ ПРОГРАМУВАННЯ

Алгоритм пошуку точного розв'язку. Ідея алгоритму полягає в тому, щоб розбивати задачу на підзадачі й вирішувати кожну крок за кроком.

ОСОБЛИВІСТЬ ЗАСТОСУВАННЯ ДИНАМІЧНОГО ПРОГРАМУВАННЯ

- Нехай n – увесь доступний час, а s – усі доступні замовлення. Тоді навіть якщо ми розіб'ємо загальну задачу на n задач, кожна з яких містить s змінних. Навіть за умови, що ми маємо 30 днів і 30 змінних, задача досі буде складною через багатовимірність.
- Також ми стикаємося з проблемою наслідків, оскільки кожний крок має вплив на наступні і пов'язаний з усіма попередніми кроками. Тобто, після виконання кожного замовлення капітал змінюється і прибуток має вплив на наступні замовлення. Водночас, капітал пов'язується з минулими замовленнями. Через це кількість необхідних обчислень сильно збільшується.
- Отже, точне рішення знайти не вийде через складність обчислення. Тому, будемо розділяти загальну задачу на такі окремі підзадачі, які буде можливо обчислити, й сумуватимемо їхні результати.



АЛГОРИТМ РОЗВ'ЯЗАННЯ

ПІДГОТОВКА

- Визначаємо оптимальне розбиття задачі по часу з урахуванням розподілу часу виконання замовлень і складності обчислень.
- Розбиваємо замовлення на категорії по часу

РЮКЗАК

- Для кожної категорії застосовуємо розв'язок задачі про рюкзак методом динамічного програмування.
- Створюємо список, який містить список оптимальних замовлень і список з прибутком, витратою по часу.
- Додаємо утворений список в словник. Ключем якого є чило, що відображає скільки замовлень і які були до того, як ми додаємо поточний.
- Видаляємо замовлення із загального списку замовлень.

ДЕРЕВО

- Повторюємо пункт 2 до тих пір, поки в нас не залишиться замовлень, або час виконання чергової партії замовлень буде менше встановленого.
- В результаті у словнику в нас будується дерево, яке складається з усіх можливих способів закупки товару партіями.

РЕЗУЛЬТАТИ

- Ми проходимо всі кінцеві вершини і знаходимо за якого набору партій буде максимальний прибуток.
- Виводимо результати, в якому порядку виконувати замовлення, які витрати капіталу/часу, прибуток будуть після кожної партії замовлень.

ПІДГОТОВКА

- Визначаємо оптимальне розбиття задачі по часу з урахуванням розподілу часу виконання замовлень і складності обчислень.
- Розбиваємо замовлення на категорії по часу

```
[[[1, 4, 300, 330], [2, 4, 400, 440], [3, 4, 500, 550], [4, 4, 600, 660],

[5, 4, 550, 605], [6, 4, 450, 540], [7, 4, 350, 420], [8, 4, 650, 813],

[9, 4, 400, 500], [10, 4, 550, 688]], [[11, 7, 350, 385],

[12, 7, 450, 495], [13, 7, 600, 720], [14, 7, 700, 840],
```

```
data = [[1, 4, 300, 330],
        [2, 4, 400, 440],
        [3, 4, 500, 550],
        [4, 4, 600, 660],
        [5, 4, 550, 605],
        [6, 4, 450, 540],
        [7, 4, 350, 420],
        [8, 4, 650, 813],
        [9, 4, 400, 500],
        [10, 4, 550, 688],
        [11, 7, 350, 385],
        [12, 7, 450, 495],
        [13, 7, 600, 720],
        [14, 7, 700, 840],
        [15, 7, 580, 725],
        [16, 7, 470, 588],
        [17, 7, 550, 715],
        [18, 7, 680, 884],
        [19, 7, 620, 837],
        [20, 7, 600, 810],
        [21, 11, 250, 275],
        [22, 11, 450, 563],
        [23, 11, 650, 813],
        [24, 11, 700, 945],
        [25, 11, 580, 783],
        [26, 11, 470, 658],
        [27, 11, 550, 770],
        [28, 11, 680, 986],
        [29, 11, 620, 899],
        [30, 11, 500, 725],
        [31, 11, 400, 600]]
CAPITAL = 3000
DAY = 30
```

РЮКЗАК

- Для кожної категорії застосовуємо розв’язок задачі про рюкзак методом динамічного програмування.
- Створюємо список, який містить список оптимальних замовлень і список з прибутком, витратою по часу.
- Додаємо утворений список в словник. Ключем якого є чило, що відображає скільки замовлень і які були до того, як ми додаємо поточний.
- Видаляємо замовлення із загального списку замовлень.

прибуток

часу витрачено

```
[[10, 4, 550, 688], [9, 4, 400, 500], [8, 4, 650, 813], [7, 4, 350, 420], [6, 4, 450, 540], [4, 4, 600, 660], [621, 4]]
```

Індекс - черговість категорій часу (номер складу)

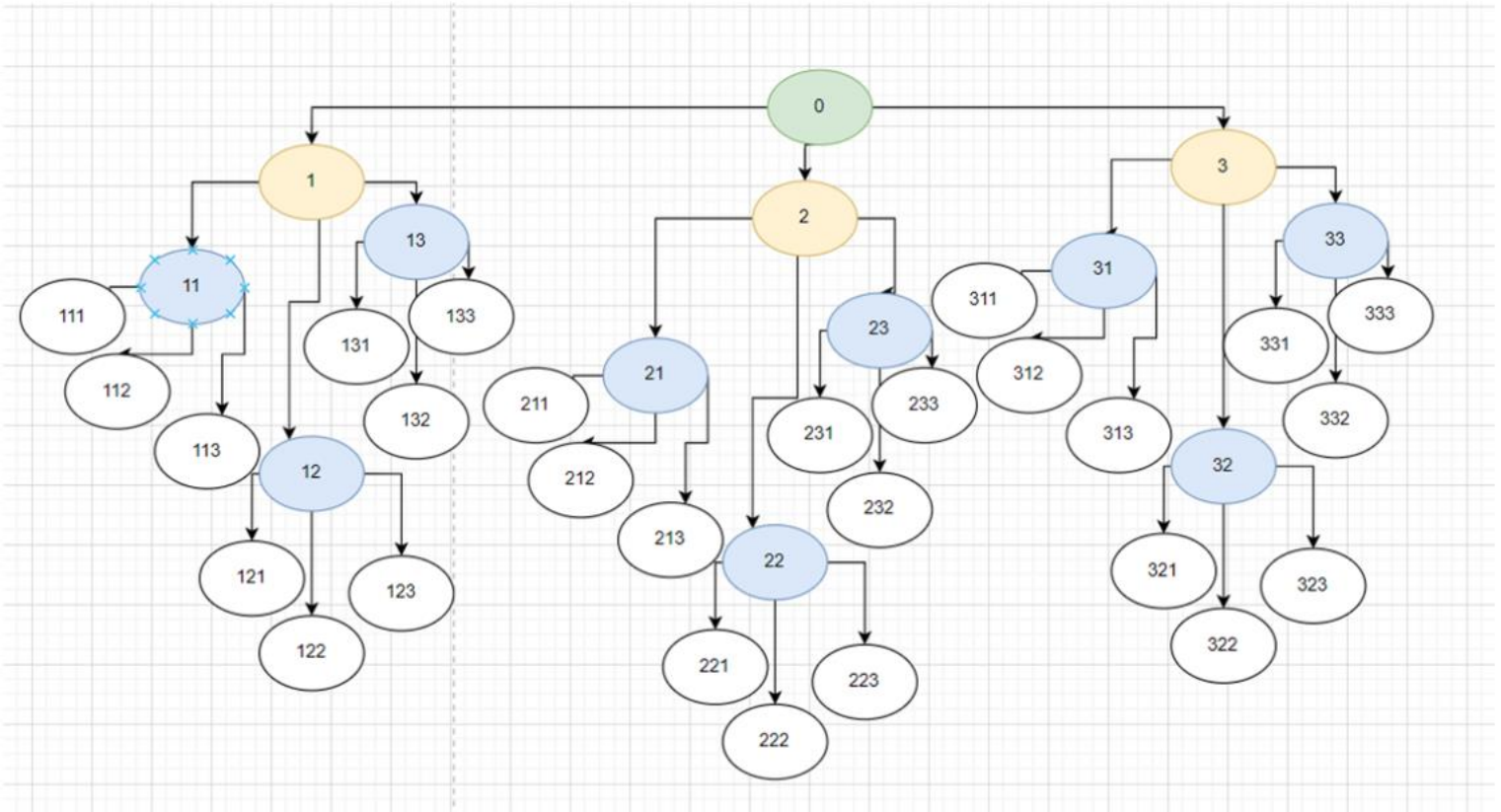
```
'1': [[10, 4, 550, 688], [9, 4, 400, 500], [8, 4, 650, 813], [7, 4, 350, 420], [6, 4, 450, 540], [4, 4, 600, 660], [621, 4]]
```

```
'11': [[5, 4, 550, 605], [3, 4, 500, 550], [2, 4, 400, 440], [1, 4, 300, 330], [796, 8]]
```

```
'112': [[20, 7, 600, 810], [19, 7, 620, 837], [18, 7, 680, 884], [16, 7, 470, 588], [15, 7, 580, 725], [1690,15]]
```

ДЕРЕВО

- Повторюємо пункт 2 до тих пір, поки в нас не залишиться замовлень, або час виконання чергової партії замовлень буде менше встановленого.
- В результаті у словнику в нас будується дерево, яке складається з усіх можливих способів закупки товару партіями.



РЕЗУЛЬТАТИ

- Ми проходимо всі кінцеві вершини і знаходимо за якого набору партій буде максимальний прибуток.
- Виводимо результати, в якому порядку виконувати замовлення, які витрати капіталу/часу, прибуток будуть після кожної партії замовлень.

```
'332': [[31, 11, 400, 600], [30, 11, 500, 725],
```

```
[3565, 29]]}
```

- Бачимо, що найоптимальніший шлях - 332 з відповідними замовленнями. Кількість часу - 29 днів. Чистий прибуток - 3565$

ДЯКУЮ ЗА УВАГУ!