**Алчни алгоритми и динамично програмиране**

(Изготвил за курса по СДП, спец. „Математика и информатика“: Михаела Папаринова, спец. „Анализ на данни“, 2023 г.)

1. **Алчни алгоритми:**

Определение: Алчните алгоритми извършват оптималния избор на всяка стъпка, като се стремят да намерят цялостния оптимален начин за решаване на целия проблем. Тези алгоритми се използват, когато локално оптимален избор води до глобално оптимално решение.

а. Проблем за монетите

Намерете минималния брой монети, чиято сума да е равна на дадена. Да приемем, че имате безкраен брой от всеки тип монети.

\*\*Пример:\*\*

- Монети: [1, 5, 10, 25]

- Обща сума: 63

\*\*Решение:\*\*

Алчният алгоритъм започва с избиране на най-голямата монета (по вид), която е по-малка или равна на оставащата сума. Този процес се повтаря до достигане на общото количество.

- Първо изберете 25 (оставащо количество = 38)

- След това изберете 25 отново (оставащо количество = 13)

- След това изберете 10 (оставащо количество = 3)

- Накрая изберете три 1 (оставащо количество = 0)

Общо използвани монети = 2 (25) + 1 (10) + 3 (1) = 6 монети

б. Проблем с избора на дейност

\*\*Проблем:\*\* Като се има предвид набор от дейности с начален и краен час, изберете максималния брой дейности, които могат да бъдат извършени от един човек, като се приеме, че човек може да работи само върху една дейност в даден момент.

\*\*Пример:\*\*

- Дейности: { (1, 2), (3, 4), (0, 6), (5, 7), (8, 9), (5, 9) }

\*\*Решение:\*\*

Алчният алгоритъм сортира дейностите въз основа на тяхното крайно време и след това избира първата дейност. След това избира следващата дейност с начален час, по-голям или равен на крайния час на предишната избрана дейност.

- Сортирайте дейностите по време на приключване: { (1, 2), (3, 4), (5, 7), (5, 9), (8, 9), (0, 6) }

- Изберете (1, 2)

- След това изберете (3, 4)

- След това изберете (5, 7)

- Накрая изберете (8, 9)

Общо избрани дейности = 4

в. Проблем за раницата

\*\*Проблем:\*\* При дадени тегла и стойности на n артикула, поставете тези артикули в раница с капацитет W, за да получите максималната обща стойност в раницата.

\*\*Пример:\*\*

- Елементи: { Стойност: 60, Тегло: 10 }, { Стойност: 100, Тегло: 20 }, { Стойност: 120, Тегло: 30 }

- Капацитет на раницата: 50

\*\*Решение:\*\*

Алчният алгоритъм изчислява съотношението (стойност/тегло) за всеки артикул и сортира всички артикули по това съотношение. След това взема предмета с най-високо съотношение и ги добавя, докато раницата не може да поеме повече.

- Изчислете съотношението стойност/тегло: {6, 5, 4}

- Сортирайте артикулите по съотношение стойност/тегло

- Вземете целия първи артикул и 40 единици от втория артикул

Максимална стойност = 60 (пълен първи елемент) + 100 \* (40/20) (част от втория елемент) = 60 + 200 = 260

**2. Разделяй и владей:**

Дефиниция: Този подход разделя проблема на по-малки, по-лесно разбираеми подпроблеми, решава всеки от тези подпроблеми само веднъж и съхранява техните решения. Това е особено ефективно, когато даден проблем може естествено да бъде разделен на подобни подпроблеми.

Пример: Merge Sort е пример за разделяй и владей. Той разделя несортиран списък на N подсписъци, всеки от които съдържа един елемент (списък от един елемент се счита за сортиран) и след това многократно обединява тези подсписъци, за да създаде нови сортирани подсписъци, докато остане само един подсписък.

**3. Динамично програмиране:**

Определение: Динамичното програмиране включва решаване на сложни проблеми чрез разделянето им на по-прости подпроблеми. Използва се, когато подпроблемите не се припокриват (т.е. не са независими)

Пример: Изчислението на последователността на Фибоначи е често срещан пример. Вместо рекурсивно изчисляване на числата на Фибоначи (което преизчислява много подпроблеми), динамичното програмиране съхранява резултатите от тези подпроблеми, за да избегне излишна работа.

**4. Алгоритми с пълно изчерпване и backtracking:**

Дефиниция: Тези алгоритми изследват всички възможни случаи, за да намерят решение. Backtracking е усъвършенствана форма на пълно изчерпване, при която алгоритъмът се опитва да изгради решение постепенно и изоставя път веднага щом установи, че този път не може да доведе до решение.

Пример: Шахматна дъска с N-царици, където целта е да поставите N царици на NxN шахматна дъска, така че две царици да не се заплашват една друга. Алгоритъмът поставя царица на дъската, след което рекурсивно се опитва да постави допълнителни царици в оставащите полета, като се връща назад, когато поставянето доведе до конфликт.

**5. Проблем с максимален подмасив:**

Проблем: Намерете подмасив на едномерен масив от числа, който има най-голяма сума на елементите.

Решение: Алгоритъмът на Kadane е ефективно решение. Той претърсва целия списък и на всяка позиция намира максималната сума на подмасива, завършващ там. Това се постига чрез поддържане на текущия брой на максималния подмасив, който завършва на всяка позиция.

**6. Алгоритъм за двоично търсене:**

Проблем: Ефективно намиране на позицията на елемент в сортиран масив.

Решение: Алгоритъмът за двоично търсене започва със сравняване на средния елемент на масива с целевата стойност. Ако целевата стойност съвпада със средния елемент, се връща позицията му в масива. Ако целевата стойност е по-малка или по-голяма от средния елемент, търсенето продължава съответно в дясната или лявата половина на масива, елиминирайки другата половина от разглеждане.

**7. Проблем с пътуващия търговец:**

Проблем: Намерете най-краткия възможен маршрут, който посещава набор от градове и се връща в града на произход.

Решение: Проблемът може да бъде решен чрез груба сила, опитвайки всички пермутации на градовете и избирайки най-кратката обиколка. Могат да се използват и алгоритъма за най-близкия съсед, при който продавачът избира най-близкия непосетен град като следващ ход.

**8. Решаване на Судоку:**

Проблем: Попълнете таблица 9×9 с цифри, така че всяка колона, всеки ред и всяка от деветте подмрежи 3×3 да съдържат всички цифри от 1 до 9.

Решение: Често срещан подход за решаване на судоку пъзели е използването на backtracking. Алгоритъмът поставя число в първата открита празна клетка, след което се премества в следващата клетка и повтаря процеса. Ако не може да се постави допустим номер в нова клетка, алгоритъмът се връща към предишната клетка и опитва различно число.

**Източници**

Проф. К. Стефанов, Лекция за Пълно изчерпване, Лакоми алгоритми, Разделяй и владей, Динамично програмиране, Търсене с връщане назад, „Структури от данни и алгортми“ за спец. „Математика и информатика“, последно посетена: 05.10.2024