Лекція 20

Рекурсивний перебір.

Загальні відомості про рекурсивний перебір

У багатьох практичних завданнях з різних предметних областей потрібно ***знайти загальну кількість варіантів рішення, число елементів в повному наборі рішень.*** Іноді, виходячи з постановки завдання, *досить знайти один з варіантів, відповідних умові завдання*. У деяких завданнях вивчається питання про існування рішення як такого.

Відповіді на ці запитання, як правило, *вимагають проведення вичерпного пошуку в деякій множині всіх можливих варіантів, серед яких знаходяться вирішення конкретного завдання*.

Існують два загальних методу організації вичерпного пошуку:

- перебір з поверненням (backtracking)
- метод решета.

Загальні відомості про рекурсивний перебір

- Рішення завдання методом перебору з поверненням будується конструктивно послідовним розширенням часткового вирішення.
- Якщо на конкретному етапі таке розширення провести не вдається, то відбувається повернення до більш короткому часткового вирішення, і спроби його розширити тривають.
- Для прискорення перебору з поверненням обчислення завжди намагаються організувати так, щоб була можливість відмовитися якомога раніше від якомога більшої кількості свідомо невідповідних варіантів.
- Незначні модифікації методу перебору з поверненням, пов'язані з поданням даних або особливостями реалізації, мають і інші назви:
  - метод гілок і меж (branch and bound),
  - пошук в глибину (depth first search),
  - метод проб і помилок.
- Перебір з поверненням практично одночасно і незалежно був винайдений багатьма дослідниками ще до його формального опису. Він знаходить застосування при вирішенні різних комбінаторних задач в області штучного інтелекту.

Загальні відомості про рекурсивний перебір

- При використанні методу решета замість конструктивної побудови рішень задачі з безлічі можливих варіантів виключаються всі елементи, які не є рішеннями. Методи решета знайшли широке застосування в теоретико-числових завданнях.
- Метод перебору з поверненням і метод решета, строго кажучи, не є ні методами, ні алгоритмами вирішення завдань. Їх слід сприймати як ***деякі загальні схеми***, які застосовуються для вирішення того чи іншого завдання. Реалізація цих схем у вигляді конкретних алгоритмів часто вимагає значних додаткових зусиль в поданні даних і описі залежностей між ними.
- З'єднання методу перебору з поверненням і рекурсії визначає специфічний спосіб реалізації рекурсивних обчислень і називається поворотної рекурсією**. *Це поєднання двох ефективних методів реалізації переборних алгоритмів***.
- При використанні поворотної рекурсії відпадає необхідність безпосередньо організовувати повернення і відслідковувати правильність їх здійснення. Вони, як правило, стають вбудованої частиною механізму виконання рекурсивних викликів.

Обчислювальна схема перебору з поверненням

- Загальна постановка класу задач, *до яких свідомо застосуємо алгоритм перебору з поверненням.*
- Нехай
  - M0, M1, ..., Mn-1 - n кінцевих лінійно впорядкованих множин
  - G - сукупність обмежень (умов), що ставлять у відповідність векторах виду, логічне значення {істина, брехня}.
  - Вектори v = (v0, v1, ..., vk) T, для яких G (v) = істина, назвемо частковими рішеннями.
  - Нехай, далі, існує конкретне правило P, відповідно до якого деякі з часткових рішень можуть оголошуватися повними рішеннями. Тоді можлива постановка наступних пошукових завдань.
    - Знайти всі повні рішення або встановити відсутність таких.
    - Знайти хоча б одне повне рішення або встановити його відсутність.
- ***Загальний метод*** вирішення наведених завдань полягає в послідовному покомпонентно нарощуванні вектора v зліва направо, починаючи з v0, і подальших перевірках його обмеженнями G і правилом P.

Обчислювальна схема перебору з поверненням

- У загальному випадку цей метод призводить до алгоритмів з *експоненційної тимчасової складністю*.
- Застосовується метод в основному до класу так званих Np-повний завдань (завдання комівояжера, задача про рюкзак і т. д.).
- Завдання цього класу еквівалентні один одному в тому сенсі, що всі вони вирішувані ***недетермінованими алгоритмами полиномиальной складності***.
- Для них відомо, що або всі вони вирішувані, або жодна з них не можна вирішити детермінованими алгоритмами полиномиальной складності.

Обчислювальна схема перебору з поверненням

- Наведемо схему виконання недетермінірованного алгоритму. Нехай алгоритм виконується до тих пір, поки не доходить до місця, з якого повинен бути зроблений вибір з кількох альтернатив.
- Детермінований алгоритм однозначно здійснить вибір конкретної альтернативи і продовжить працювати відповідно до ці вибором.
- Недетермінований алгоритм досліджує всі можливості одночасно, як би копіюючи себе для реалізації обчислень по всіх альтернатив одночасно. Далі все копії працюють незалежно один від одного і в міру необхідності продовжують створювати нові копії. Копія, яка зробила неправильний або безрезультатний вибір припиняє свою роботу. Копія, знайшла рішення задачі, має засвідчити це, даючи тим самим сигнал іншим копіям про припинення обчислень. Недетермінірованние алгоритми, будучи вельми корисною і продуктивною абстракцією, рекурсивні по суті, бо при реалізації оператора вибору фактично звертаються самі до себе.
- .

Завдання про розстановку ферзів на шахівниці

- *Складіть рекурсивную функцію, яка знаходить можливу розстановку n ферзів на шахівниці розміром nxn так, щоб вони не били один одного (n - натуральне число).*
- Відповідно до загальної схеми алгоритму перебору з поверненням запропоновану задачу можна вирішувати за алгоритмом "Всі розстановки", **але** завершити виконання алгоритму при знаходженні першої необхідної розстановки або при отриманні висновку про неможливість отримати потрібну комбінацію. Відповідно до алгоритму ферзі розставляються послідовно на вертикалях з номерами від нуля і далі. В процесі виконання можливі зняття ферзів з дошки (***повернення***).

Завдання про розстановку ферзів на шахівниці

Алгоритм "Всі розстановки"

Крок 1. Вважаємо (D - безліч рішень, j - поточний стовпець для чергового ферзя).

Крок 2. Намагаємося в стовпці j просунути вниз по вертикалі або новий (якщо стовпець j порожній), або вже наявний там ферзь на найближчий допустимий рядок. Якщо це зробити не вдалося, то переходимо до кроку 4.

Крок 3. Збільшуємо j на 1, тобто переходимо до наступного колонки. Якщо j <n-1, то переходимо до кроку 2. В іншому випадку j = n-1, тобто все вертикалі вже зайняті. Знайдене часткове вирішення запам'ятовуємо в множині D і переходимо до кроку 2.

Крок 4. Зменшуємо j на 1, тобто знімаємо ферзь з стовпця j і переходимо до попереднього колонки. Якщо j> 0, то виконуємо крок 2. Інакше обчислення припиняємо. Рішення завдання знаходяться в множині D, яке, може бути і порожнім.

Рішення завдання для невеликих розмірів дошки можна знайти "вручну", але для розробки алгоритму необхідно провести моделювання умови і зупинитися на якомусь поданні даних.

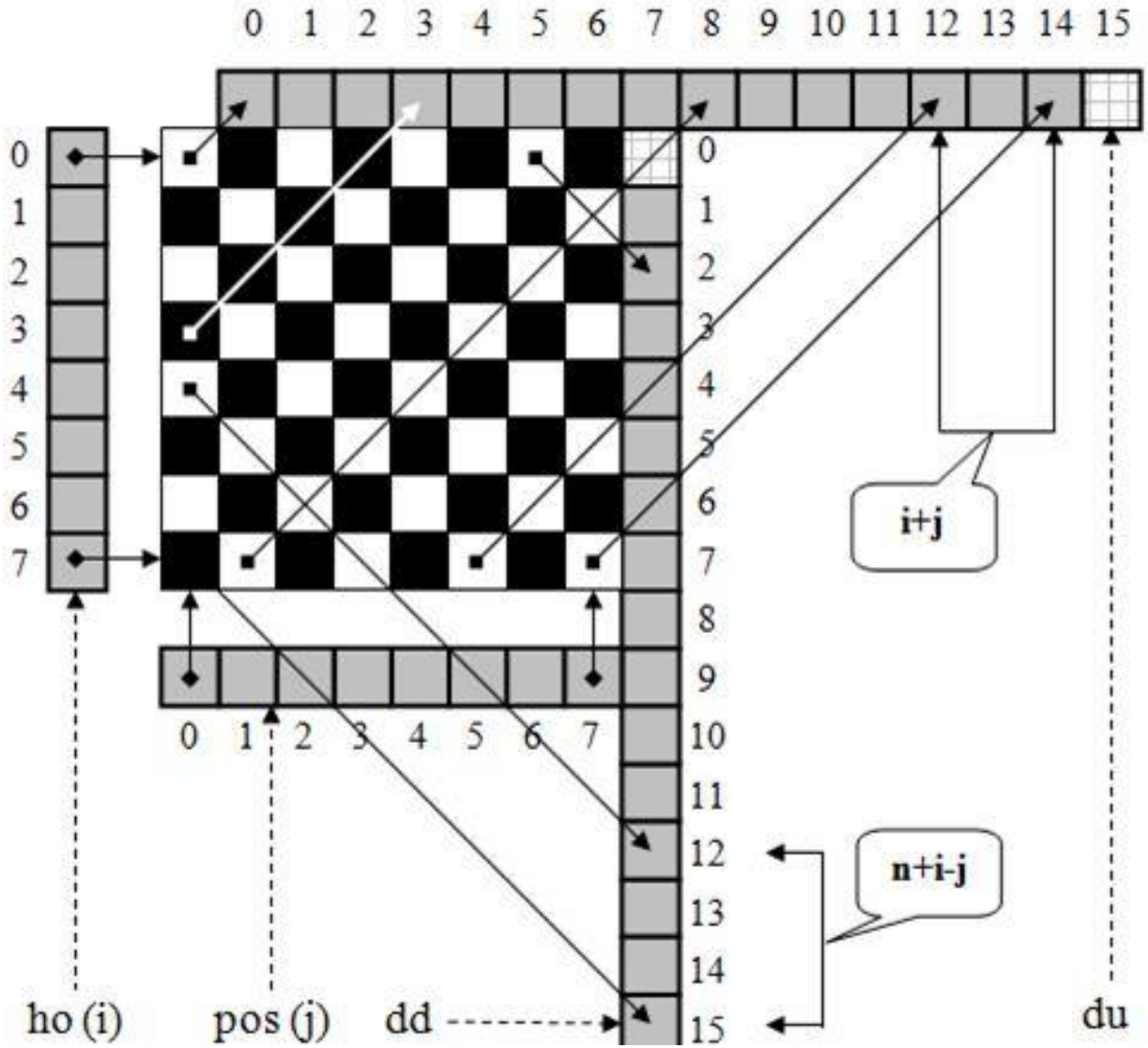
Завдання про розстановку ферзів на шахівниці

- Проведемо параметризацію завдання. Введемо чотири допоміжних вектора: pos, ho, dd і du с довжинами n, n, 2n-1 і 2n-1 відповідно.

Використовувати їх будемо в такий спосіб

- ho_i = 1, якщо на горизонталі з номером i (i = 0,1, ... n-1) є ферзь, і ho_i = 0 - в іншому випадку;
- du_s = 1, якщо на діагоналі з номером s (s = 0,1, ... 2n-2), що йде зліва направо і знизу вгору, є ферзь, і dus = 0 - в іншому випадку;
- dd_s = 1, якщо на діагоналі з номером s (s = 0,1, ... 2n-1), що йде зліва направо і зверху вниз, є ферзь, і dds + 1 = 0 - в іншому випадку;
- pos_j = i, якщо в позиції (i, j) (i, j = 0,1, ... n-1) є ферзь.

Завдання про розстановку ферзів на шахівниці



Використання цих правил дозволяє отримати такі твердження:

У позицію (i, j) можна поставити ферзь, якщо ho_i + du_i + j + dd_n + i-j = 0.

Поставити ферзь в позицію (i, j) є присвоювання: ho_i = 1, du_i + j = 1, dd_n + i-j = 1.

Прибрати ферзь з позиції (i, j) є присвоювання: ho_i = 0, dui + j = 0, ddn + i-j = 0.

Завдання про розстановку ферзів на шахівниці

- Даний опис алгоритму є моделлю рішення загальної задачі про знаходження всіх варіантів розстановок.
- Рекурсія тут здійснюється за не зовсім стандартною схемою. У кожному рекурсивному виклиці глибини j робиться спроба помістити ферзя в деяку позицію i стовпця j (i, j = 0,1, ... n-1), а сам виклик відповідає переходу від роботи з поточним стовпцем до роботи з наступним стовпцем.

Завдання про розстановку ферзів на шахівниці

- При цьому на початку обчислень і при переходах до будь-якого подальшого рекурсивному викликом параметр і змінюється від нуля і далі з кроком, 1, намагаючись прийняти значення найменшого номера поля, допустимого для установки ферзя. При переходах до будь-якого попереднього рекурсивному викликом параметр і продовжує змінюватися від свого поточного на даному рівні значення з кроком 1, також намагаючись прийняти значення найменшого номера поля, допустимого для установки ферзя. Якщо в поточному стовпці ферзь встановити вже не вдається, то ситуацію, що склалася назвемо тупиком. Попадання в глухий кут призводить до завершення поточного рекурсивного виклику, тобто до повернення до попередньої колонки і продовження роботи з нею. Інших випадків завершення рекурсивних викликів не існує.
- Тому базою рекурсії ми повинні вважати сукупність всіх тупиків. Зауважимо, що в даному випадку елементи бази заздалегідь до обчислень невідомі.
-
- Після установки ферзя в одному з полів і останнього стовпця i = n-1 формується одне з рішень задачі - при пошуку одного варіанта розстановки на цьому етапі слід завершити виконання алгоритму. При пошуку всіх розстановок обчислення припиняються, коли ми потрапляємо в глухий кут при роботі зі стовпцем 0. Отримані рішення задачі, якщо вони є, повертаються у вигляді стовпців матриці otv, починаючи від першого і далі.

Завдання про основні розподіли

Для формулювання наступного завдання введемо поняття. Серед усіх розстановок n ферзів на дошці nxn відділімо ОКРЕМІ непересічні класи Hs (s = 0,1, ..., q) розстановок так, що всі елементи даного класу можна отримати з будь-якого его представника будь-якими елементарними перетвореннями типу:

- поворот дошки в її площині коло центру на 90, 180 и 270;
- перетворення симетрії відносно діагоналей;
- перетворення симетрії відносно прямих, що проходять через центр дошки по межах клітин.

Взявши по одному представнику з класу Hs (s = 0,1, ..., q), одержимо деяку множину, що називається основними розстановками. *Скласти рекурсивну функцію, яка знаходить будь-яку безліч основних розстановок n ферзів на шахівниці розміру nxn.*

Завдання про основні розподіли

Це завдання вірішується за допомогою рекурсивних функцій практично аналогічно завдання з розстаовкою ферзів.

Відмінності тут Такі. Рекурсивна функція послідовно формує множину з можливих розстановок ферзів на дошці, проте НЕ всі з них запам'ятовуються в матриці ВІДПОВІДІ otv.

Чергова отримана розстановка піддається перевірці - включать чи не включать її в матрицю otv. Робиться це в такий спосіб. З вектора pos елементарних перетвореннь формуються ще сім розстановок (деякі з них можут збігатися).

Якщо жодна з них не входить в потокову матрицю відповідей otv, то otv доповнюється новим рішенням і так далі. Отже, по завершенню обчисленнь otv буде містіти деяк безліч основних розстановок.

Задача про рюкзак

- ***Задача:***
- Завдання про рюкзак
- Дано N предметів, ni предмет має масу wi> 0 і вартість pi> 0.
- Необхідно вибрати з цих предметів такий набір, щоб сумарна маса не перевищувала заданої величини W (місткість рюкзака), а сумарна вартість була максимальна.

Задача про рюкзак

- ***Формулювання задачі***
- Дано:
- N предметів,
- W - місткість рюкзака, w = {w1, w2, ..., wN} - відповідний йому набір додатних цілих ваг,
- p = {p1, p2, ..., pN} - відповідний йому набір додатних цілих вартостей.
- Потрібно знайти набір бінарних величин B = {b1, b2, ..., bN}, де bi = 1, якщо предмет ni включений в набір, bi = 0, якщо предмет ni не включений, і такий що:
- 1. $b1w1 + \cdots + bNwN \leqslant W$
- 2. b1p1 + ··· + bNpN максимальна.

Задача про рюкзак

- ***Варіанти розв'язків***
- Завдання про рюкзаку можна вирішити кількома способами:

1. Перебирати все підмножини набору з N предметів.
   Складність такого рішення O (2N)
2. Методом Meet-in-the-middle.
   Складність рішення O (2N / 2N)
3. Метод динамічного програмування.
   Складність - O (NW)

Задача про рюкзак

Приклад
W=13,N=5
w1=3,p1=1 w2=4,p2=6 w3=5,p3=4 w4=8,p4=7 w5=9,p5=6
Числа від 0 до 13 в першому рядку позначають місткість рюкзака.
У першому рядку як тільки місткість рюкзака n⩾3 додаємо в рюкзак 1 предмет.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
k = 0	0	0	0	0	0	0	0	0	0	0	0	0	0
k = 1	0	0	1	1	1	1	1	1	1	1	1	1	1
k = 2	0	0	1	6	6	6	7	7	7	7	7	7	7
k = 3	0	0	1	6	6	6	7	7	10	10	10	11	11
k = 4	0	0	1	6	6	6	7	7	10	10	10	13	13
k = 5	0	0	1	6	6	6	7	7	10	10	10	13	13

Задача про рюкзак

- Розглянемо k = 3, при кожному s⩾5 (так як w3 = 5) порівнюємо A [k-1] [s] і A [k -1] [s-w3] + p3 і записуємо в A [k] [s] вартість або рюкзака без третього предмета, але з такою ж вагою, або з третім предметом, тоді вартість дорівнює вартості третього предмету плюс вартість рюкзака з місткістю на w3 менше.
- Максимальна вартість рюкзака знаходиться в A (5,13).
- Відновлення набору предметів, з яких складається максимально дорогий рюкзак.
- Починаючи з A (5,13) відновлюємо відповідь. Будемо йти в зворотному порядку по k.

Задача про рюкзак

- Червоним позначено наш шлях
- Таким чином, в набір входить 2 і 4 предмет.
- Вартість рюкзака: 6 + 7 = 13
- Вага рюкзака: 4 + 8 = 12

	1	2	6	4	5	6	7	8	9	10	11	12	13
k = 0	0	0	0	0	0	0	0	0	0	0	0	0	0
k = 1	0	**0**	1	1	1	1	1	1	1	1	1	1	1
k = 2	0	0	1	**6**	6	6	7	7	7	7	7	7	7
k = 3	0	0	1	**6**	6	6	7	7	10	10	10	11	11
k = 4	0	0	1	6	6	6	7	7	10	10	10	13	**13**
k = 5	0	0	1	6	6	6	7	7	10	10	10	13	**13**

Дякую за увагу!