Лекція 5. Покращені алгоритми внутрішнього сортування

Покращення алгоритму бульбашкового сортування

Кролики і черепахи. Позиція елементів, що підлягають сортуванню відіграє велику роль у питанні продуктивності даного алгоритму. Великі елементи на початку списку не являють проблеми, оскільки вони досить швидко переміщаються на свої місця. Однак, малі елементи у кінці списку переміщаються на його початок дуже повільно. Це призвело до того, що обидва типи елементів було названо кроликами і черепахами, відповідно.

З метою підвищення швидкодії алгоритму, у свій час було здійснено чимало зусиль для зменшення кількості "черепах". Сортування перемішуванням є порівняно не поганим, однак, усе ще у своєму найгіршому випадку має складність $O(n^2)$. Сортування гребінцем спершу порівнює великі елементи один з одним, а вже тоді поступово переходить до все менших і менших. Його середньостатистична швидкість приблизно рівна такій в алгоритмі Швидке сортування.

Сортування змішуванням – один із різновидів алгоритму сортування бульбашкою. Відрізняється від сортування бульбашкою тим, що сортування відбувається в обох напрямках, міняючи напрямок при кожному проході. Даний алгоритм лише трішки складніший за сортування бульбашкою, однак, вирішує так звану проблему "черепах".

Швидкодія

Ефективність алгоритму рівна $O(n^2)$ водночас для середнє статистичного та найгіршого випадку, однак, вона прямує до $O(n)$ якщо список вже не погано відсортований, наприклад, якщо кожен елемент знаходиться у позиції, яка відрізняється від кінцевої більше, ніж на k ($k \geq 1$), то його швидкодіє рівна $O(k * n)$.

Відмінності від сортування бульбашкою

Сортування змішуванням мало чим відрізняється від сортування бульбашкою. Єдина його відмінність у тому, що замість багаторазового проходження через список знизу вгору, він проходить по черзі знизу вгору і згори вниз. Він може досягати трохи вищої ефективності, ніж алгоритм сортування бульбашкою. Причиною цьому є те, що алгоритм сортування бульбашкою проходить по списку лише в одному напрямі, а тому за одну ітерацію елементи списку можна перемістити лише на один крок.

Наприклад, для того, щоб відсортувати список (2,3,4,5,1), алгоритму сортування змішуванням достатньо лише одного проходу, у той час, як алгоритму сортування бульбашкою знадобиться чотири проходи. Однак, один прохід сортування змішуванням слід рахувати за два проходи сортування бульбашкою. Зазвичай, сортування змішуванням удвічі швидше за сортування бульбашкою.

Іншою можливою оптимізацією є запам'ятовування попередніх перестановок. У наступній ітерації, перестановки не повторюватимуться, а тому алгоритм матиме коротші проходи по списку.

Приклад на Паскалі:

```
procedure bubble;
var i, j, t : byte;
begin
    for i :=2 to N do
        for j:=N down to i do
            if x[i-1]>x[j] then
                begin t:=x[j-1];x[j-1]:=x[j];x[j]:=t; end;
end;
end;
```

Сортування гребінцем – спрощений алгоритм сортування, розроблений Влодеком Добошєвічем (Wlodek Dobosiewicz) у 1980 році, і пізніше заново слідженим та популяризованим Стефаном Лакеєм (Stephen Lacey) та Річардом Боксом (Richard Box), котрі написали про нього в журналі Byte Magazine у квітні 1991 р. Сортування гребінцем є поліпшенням алгоритму сортування бульбашкою, і конкурує у швидкодії з алгоритмом Швидке сортування. Основна його ідея полягає в тому, щоб усунути так званих "черепах", або малих значень ближче до кінця списку, оскільки у сортування бульбашкою вони сильно уповільнюють процес сортування. (Кролики та великі сортування на початку списку у сортуванні бульбашкою не являють собою проблеми).

У сортуванні бульбашкою коли два елементи порівнюються, вони зажди мають розрив (відставнь один від одного) рівну 1. Основна ідея сортування гребінцем полягає у тому, що цей розрив може бути більший одиниці. (Алгоритм Сортування Шелла також базується на даній ідеї, однак, він є модифікацією алгоритму сортування вставками, а не сортування бульбашкою).

Розрив починається зі значення, що рівне довжині списку, поділеного на фактор зменшення (за звичай, 1.3; див. нижче), і список сортується з урахуванням цього значення (при необхідності воно заокруглюється до цілого). Потім розрив знову ділиться на фактор розриву, і список продовжує сортуватись з новим значенням, процес вродовжується до тих пір, доки розрив рівний 1. Далі список сортується з розривом рівним 1 до тих пір, доки не буде повністю відсортований. Таким чином, фінальний етап сортування аналогічний такому ж у сортуванні бульбашкою, однак, до цього "черепах" усувається.

Фактор зменшення

Фактор зменшення справляє великий ефект на швидкість алгоритму слортування гребінцем. В оригінальній статті, автор пропонує значення 1.3 після багатьох експериментів з іншими значеннями.

Текст описує процес вдосконалення алгоритму використовуючи значення $1/\left(1-\frac{1}{e^{\varphi}}\right)\approx 1.247330950103979$ в якості фактора зменшення. Вона також містить приклад використання алгоритму на псевдокоді.

Приклади реалізації на різних мовах програмування

Псевдокод

```
function combsort11(array input)
    gap := input.size

    loop until gap <= 1 and swaps = 0
        if gap > 1
            gap := gap / 1.3
            if gap = 10 or gap = 9
                gap := 11
            end if
        end if
        i := 0
        swaps := 0
        loop until i + gap <= input.size
            if input[i] > input[i+gap]
                swap(input[i], input[i+gap])
                swaps := 1
            end if
            i := i + 1
        end loop
    end loop
end function
```

Код на C++

```
void sort( data *array, dword size )
{
  if (!array || !size)
    return;

  dword jump = size;
  bool swapped = true;

  while (jump > 1 || swapped)
  {
    if (jump > 1)
      jump = (dword)(jump / 1.25);
    swapped = false;
    for (dword i = 0; i + jump < size; i+=jump)
      if (array[i] > array[i + jump])
        swap(array, i, i + jump), swapped = true;
  }
}
```

Покращені методи сортування

Сортування Шелла

Сортування Шелла – це алгоритм сортування, що є узагальненням сортування вставкою.

Алгоритм базується на двох тезах:

– Сортування включенням ефективне для майже впорядкованих масивів.

– Сортування включенням неефективне, тому що переміщує елемент тільки на одну позицію за раз.

Тому сортування Шелла виконує декілька впорядкувань включенням, кожен раз порівнюючи і переставляючи елементи, що знаходяться на різній

відстані один від одного.

Сортування Шелла не є стабільним.

Історія

Сортування Шелла названо начесть автора — Дональда Шелла, який опублікував цей алгоритм у 1959[1] році. В деяких пізніших друкованих виданнях алгоритм називають *сортуванням Шелла-Мацнера*, за ім'ям Нортона Мацнера. Але сам Мацнер стверджував: «Мені не довелось нічого робити з цим алгоритмом, і моє ім'я не має пов'язуватись з ним».[2]

Ідея алгоритму

На початку обираються m-елементів: $d_1, d_2, \ldots, d_m$, причому, $d_1 > d_2 > \ldots > d_m = 1$.

Потім виконується m впорядкувань методом включення, спочатку для елементів, що стоять через d_1, потім для елементів через d_2і т. д. до $d_m = 1$.

Ефективність досягається тим, що кожне наступне впорядкування вимагає меншої кількості перестановок, оскільки деякі елементи вже стали на свої місця.

Псевдокод

Сам алгоритм не залежить від вибору m і d, тому будемо вважати, що вони задані.

$Shell\_Sort(A, N)$

1. **for** $k \leftarrow 1$ **to** m
2. **do for** $i \leftarrow d[k] + 1$ **to** N
3. **do** $a \leftarrow A[i]$
4. $j \leftarrow i$
5. **while** $j - d[k] \geq 1$ **і** $A[j - d[k]] > a$
6. **do** $A[j] \leftarrow A[j - d[k]]$
7. $j \leftarrow j - d[k]$
8. $A[j] \leftarrow a$

Коректність алгоритма

Оскільки $d_m = 1$то на останньому кроці виконується звичайне впорядкування включенням всього масиву, а отже кінцевий масив буде впорядкованим.

Час роботи

Час роботи залежить від вибору значень елементів масиву d. Існує декілька підходів вибору цих значень:

- При виборі $d_1 = \left\lfloor \frac{N}{2} \right\rfloor, d_2 = \left\lfloor \frac{d_1}{2} \right\rfloor, d_3 = \left\lfloor \frac{d_2}{2} \right\rfloor, \ldots, d_m = 1$ час роботи алгоритму, в найгіршому випадку, є $O(N^2)$.
- Якщо d — впорядкованний за спаданням набір чисел виду $\frac{3^j - 1}{2}, j \in \mathbb{N}, d_i < N$, то час роботи є $O\left(N^{\frac{3}{2}}\right)$.
- Якщо d — впорядкованний за спаданням набір чисел виду $2^i 3^j, i, j \in \mathbb{N}, d_k < N$, то час роботи є $O(N \log^2 N)$.

Приклад роботи

Проілюструймо роботу алгоритму на вхідному масиві A = (5,16,1,32,44,3,16,7), d = (5,3,1).

1. Масив після впорядкування з кроком в 5: (3,16,1,32,44,5,16,7) — зроблено 1 обмін.

2. Масив після впорядкування з кроком 3: (3,7,1,16,16,5,32,44) — зроблено 3 обміна.

3. Масив після впорядкування з кроком 1: (1,3,5,7,16,16,32,44) — зроблено 5 обмінів.

Отже, весь масив впорядковано за 8 операцій обміну.