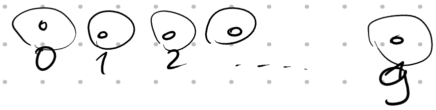
Структура данных для хранения системы непересекающихся множеств.

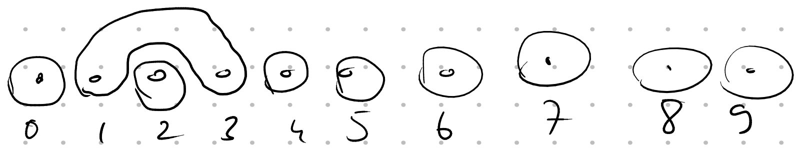
1. Создать множество: create(10)



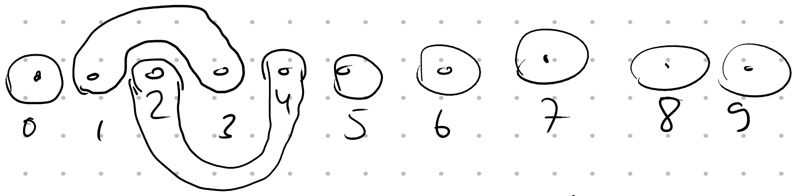
мн-во из 10 эл-тов и сразу 10 подмножеств, каждое из 1 элемента.

2. объединить 2 подмножества.

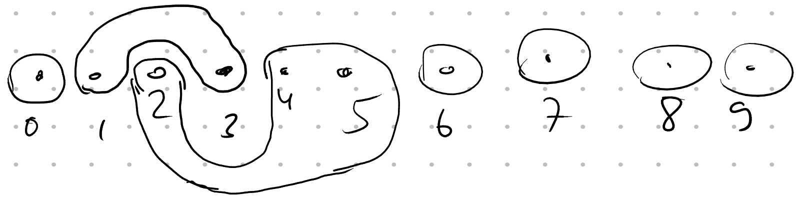
union(1,3):



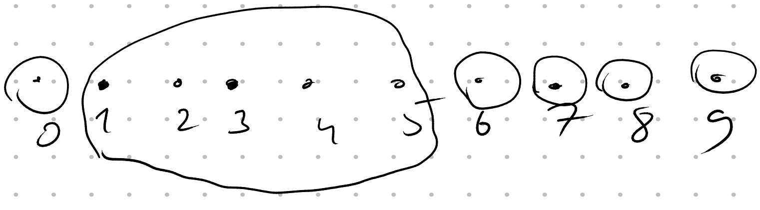
union(2,4)



union(2,5) – подмножество с 2 ∪ с 5



union(3,5)

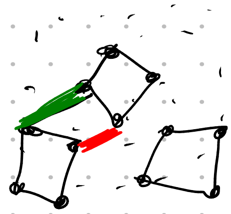


3. find(x) – для элемента x находить "представителя" множества x, какой-то фиксир. эл-т мн-ва, в котором есть x

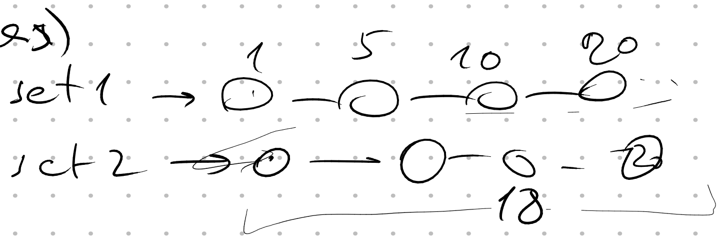
$x, y \in U_i$ (оба в одном множестве)

$\Leftrightarrow$ find(x) = find(y)

Зачем: Поиск компонент связности



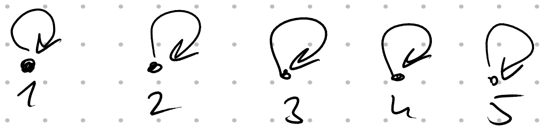
Реализация (медленная)



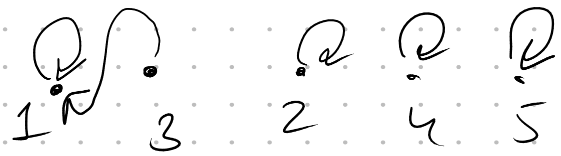
Реализация $\boxed{\text{union-find}}$ forest

Идея: храним для каждого эл-та мн-ва ссылку на другой эл-т из его же подмн-ва.

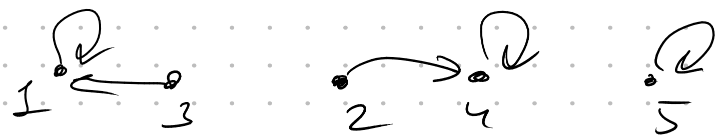
При создании графа create(1..5)



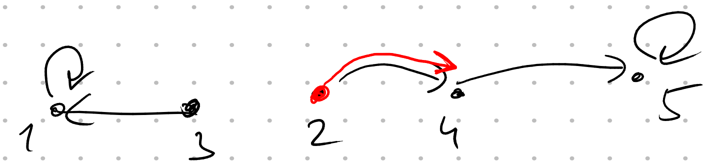
union(1,3)



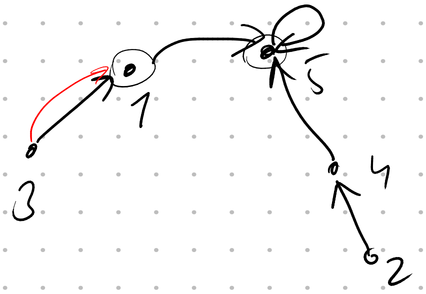
union(2,4)



union(2,5)



union(3,5)



Операция $\underline{find}(x)$ идти к корню из x.

$find(3) = 5$ $find(2) = 5$ $find(7) = 7$

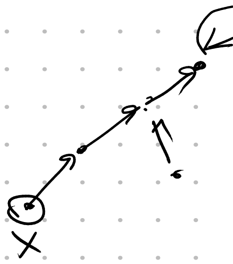
псевдокод:

$a[x]$ – куда указывает эл-т x

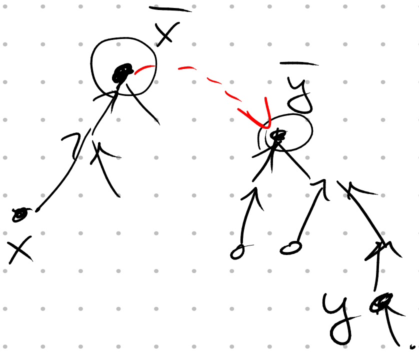
create(n) = a = new массив [0..n-1]

for i : a[i]=i

каждый показ. на себя



```
find(x):
    while a[x] ≠ x
        x = a[x]
    return x
```



```
union(x, y):
    x̄ = find(x)
    ȳ = find(y)
    if x̄ == ȳ
        return "x и y уже и так вместе"
    a[x̄] = ȳ
    return ȳ   ← вернуть представителя множества.
```

Насколько это эффективно??

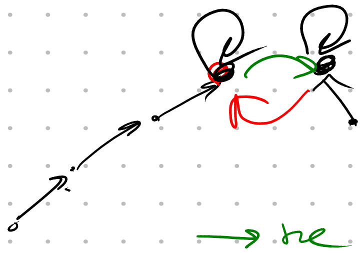
линейный поиск...

Эвристики 1. сжатие пути

find (x)

нужно все встреченные на пути эл-ты направить в $\bar{X}$.

Эвристика 2. union:

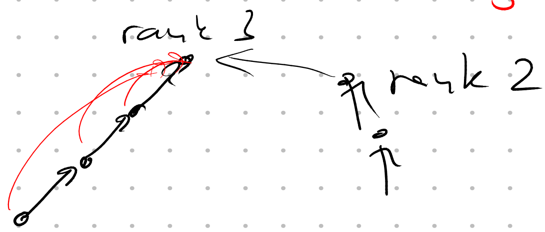


→ не выгодно

→ выгодней

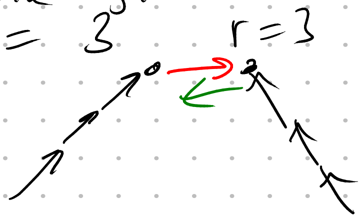
храним rank[x] := 1

(примерная высота)



при объед меньший rank → больший rank.

если rank одинак.



⇄ неважно, но rank++ (4)

<u>Утв.</u> Сложность с двумя эвристиками:

$$O(\underbrace{n\alpha(n)}_{\le 45})$$ где $\alpha(n)$ — очень медл. ф-ция

$$\approx O(n)$$ $$\alpha(n) = A^{-1}(n)$$

Динамическое программирование

Схема работы динамического алгоритма:

надо вычислить $F(n)$

1. $F(0)$ = задаём
2. $F(k)$ вычисляем по $F(0), F(1), \ldots, F(k-1)$
и сохраняем результат в массиве $f[k] := \ldots$

... это вычисление рекуррентной ф-ии.

Задача 1. Набрать сумму монетами.

Дана монетная система $= \{1, 2, 5, 10\}$

надо набрать 100 рублей

Варианты вопросов: (1) – можно ли набрать?

(2) – сколько min надо монет?

$99 = \underline{10+10+\ldots+} \, 5+2+2 \leftarrow 12$ шт.

(3) – как набрать min кол-во монет?

не оч. простая задача

$c = \{7, 11, 47\}$ можно ли 100?

жадный алг. всегда брать max монету

– в общем случае не работает

$\{1, 5, 6\}$ набрать 10 $\quad 10 = 5+5$

жадно: $10 = 6+1+1+1+1$

как решать

$F(i)$ – можно ли набрать сумму i заданной монетной системой?

$c = \{2, 5, 7\}$

$f(0) = true$ $f(1) = false$ $f(2) = true$ $f(3) = false$

$f(4) = true$ ($2+2$) $f(5) = true$ $f(6) = true$ ($2+2+2$) ...

$c_1 \dots c_k$ — номиналы монет

Вычисление $F[0] := true$ (0 всегда можно)

$F[1,2,\dots] = false$

```
for i = 1 to n
    // можно ли набрать i ?
    for j=1 to k
        если f(i - c_j)
        then f(i) = true!!
            break
```

набираем $i - c_j$
добавляем c_j
получаем i

Ответ: return $f[n]$

Пример. $c = \{2, 3, 7\}$ $n = 11$

f	0	1	2	3	4	5	6	7	8	9	10	11
	t	f	t	t	t	t	t	t	t	t	t	t

(2) Сколько min монет нужно?

$F(i)$ = сколько min нужно монет, чтобы набрать i или $+\infty$, если нельзя набрать i

$F(0) = 0$

Аналогичный алгоритм.

$$F(i) = 1 + \min\{F(i-c_1), F(i-c_2), \dots, F(i-c_k)\}$$

$c = \{2, 3, 7\}$ $n = 11$

0	1	2	3	4	5	6	7	8	9	10	11
0	∞	1	1	2	2	2	1	3	2	2	3

(3) Как именно набрать оптимально?

подход 1. Достаточно решения (2)
восстановления пути через f из (2)

подход 2

$F(i) =$ таблица (вектор.) сколько взять монеток $c_1\ c_2 \ldots c_k$.

$$F(0) = \left[\begin{array}{l} c_1 : 0 \\ c_2 : 0 \\ \vdots \\ c_k : 0 \end{array}\right] \qquad F(1) = \left[\begin{array}{l} c_1 : \_\_ \\ c_2 : \_\_ \\ \vdots \\ c_k : \_\_ \end{array}\right]$$

F	0	1	2	3	4	5	6	7	8	9	10	11
2	0											2
3	0											0
7	0					+1						1

Сложность $O(n \times k)$

Оптимизация хранить надо только $c_k + 1$ последних значений f

Задача. Наибольшая возрастающая подпоследовательность

дан массив чисел. 6 10 2 5 9 7 8 3

найти длину самой длинной последовательности

ind. $a[ind[i]] \le a[ind[i+1]]$

длина ответа: 4

Решение 1 $F(i)$ = длина наибольшей возраст. подпоследовательности, в массиве $a[1..i]$, где последний эл-т в позиции i

	6	10	2	5	9	7	8	3
F	1	2	1	2	3	3	4	2
ind	0	1	2	3	4	5	6	7

$F(0) := 1$. $F(1)$ $\boxed{6\ 10}$ $F(2)$ $\boxed{6\ 10\ 2}$ конец

$F(3)$ $\boxed{6\ 10\ 2\ 5}$ $F(4) =$ $\boxed{6\ 10\ 2\ 5\ 9}$ 2 2 3

$F(7) =$

Ответ: ищем $\max f = 4$

$$F(i) = \underbrace{F(\overset{j}{0}) \ldots F(\overset{j}{1}) \ldots\ldots F(\overset{j}{i-1})}_{\text{оставляем только те, где } a[i] \geq a[j]}$$

$$1 + \underbrace{\qquad\qquad\qquad}_{\max}$$

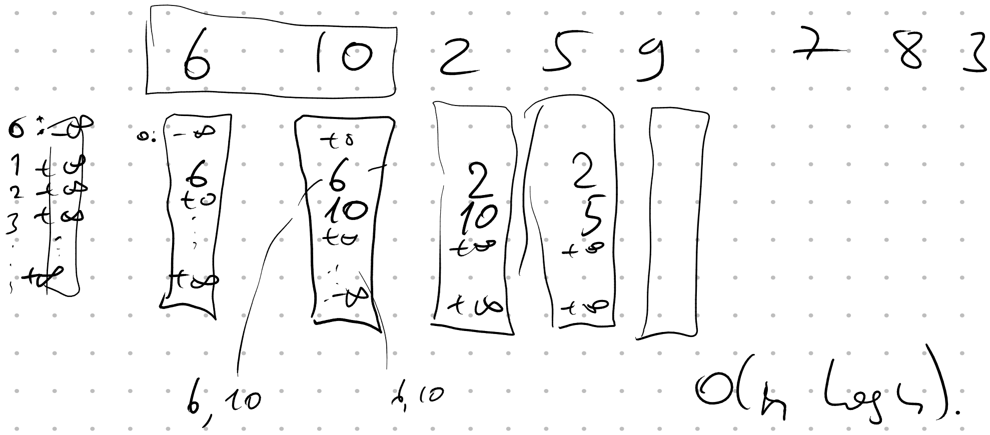
Окончательно: $\max F(i)$

Сложность $O(n^2)$

$$1+2+3+\ldots+n = \frac{n(n+1)}{2} \approx \frac{1}{2}n^2 \quad O(n^2)$$

Другая ф-ия, которая даёт сложность $O(n \log n)$

$$F(i) = \begin{bmatrix} 0: \\ 1: \\ k: \\ n: \end{bmatrix} \leftarrow$$ среди всех ↑ последовательностей длины k в массив $a[0..i]$ min конец записываем.



$O(n \log n)$.