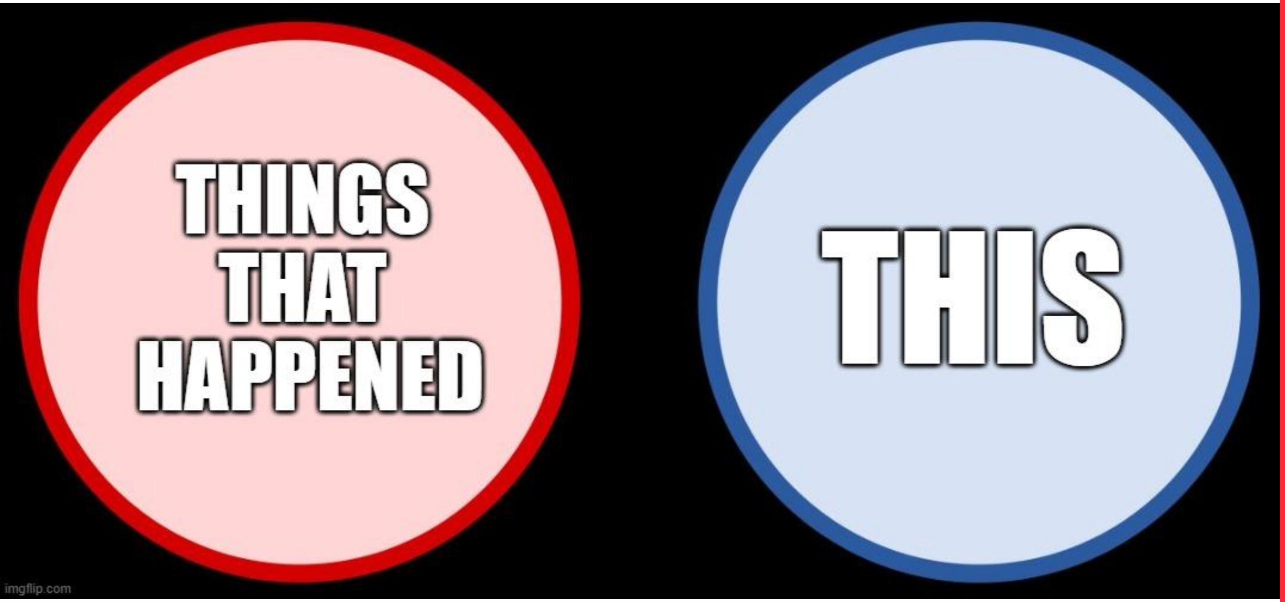
академия больших данных mail.ru group made

Графы – 3. Остовные деревья.

Шовкопляс Григорий

Введение в алгоритмы и структуры данных



Система непересекающихся множеств

Система непересекающихся множеств

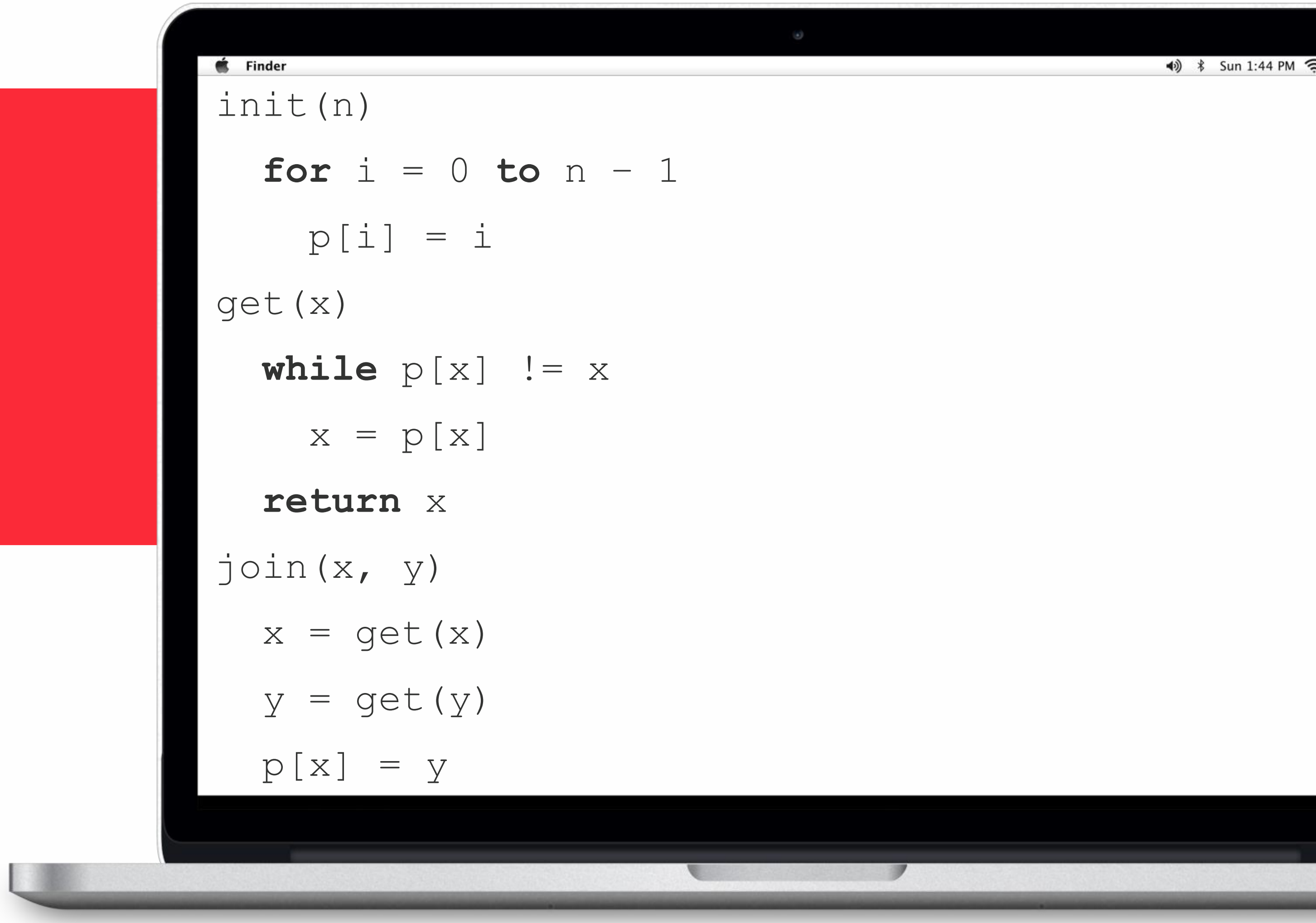
- Структура данных
- Изначально все элементы в разных множествах
- Операции
    - join(x, y): объединить множества, содержащие x и y
    - get(x): найти представителя множества, в котором лежит x
- Все работает «быстро»

Система непересекающихся множеств

- Храним в виде леса
- Для каждого элемента знаем предка
    - get(x): найти корень дерева
    - join(x, y): подвесить корень одного дерева к другому
- Все работает «небыстро»
- O(n) на операцию

Система непересекающихся множеств

- Наивная реализация
- Действительно O(n)




```
init(n)
  for i = 0 to n - 1
    p[i] = i
get(x)
  while p[x] != x
    x = p[x]
  return x
join(x, y)
  x = get(x)
  y = get(y)
  p[x] = y
```

Система непересекающихся множеств

- Время эвристик:
  - Ранговая эвристика
    - Ранг высота (пока что) дерева
    - Подвешиваем менее глубокое дерево к более глубокому
  - Эвристика сжатия путей
    - Если уж пошли до корня всех по пути переподвесим
- Обе эвристики по отдельности дают O(logn) на запрос
- Вместе $\alpha(m, n)$ – обратная функция Аккермана (меньше 5 для всех разумных чисел)

Система непересекающихся множеств

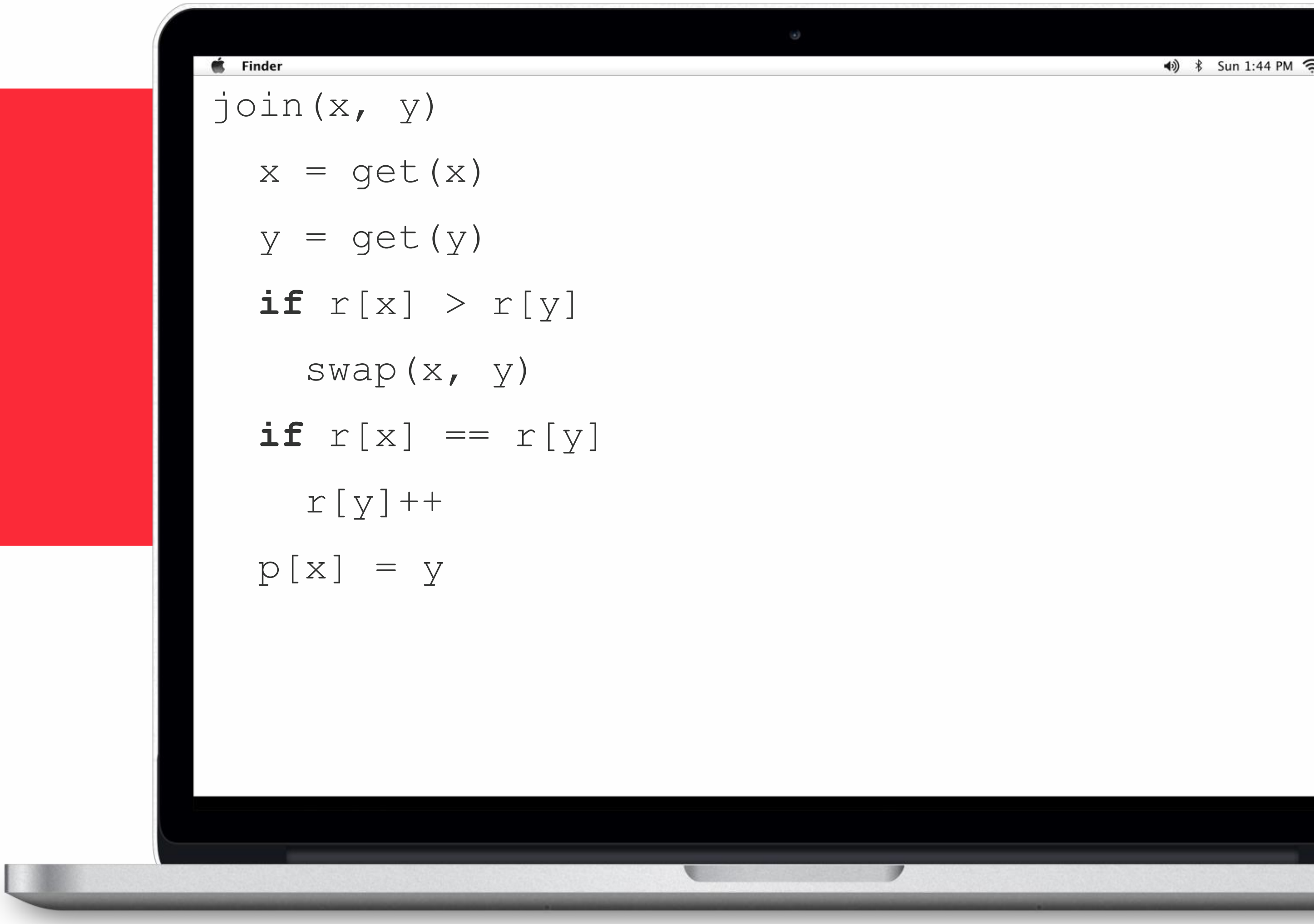
- Применим эвристики

```
init(n)
  for i = 0 to n - 1
    p[i] = i
    r[i] = 0

get(x)
  if p[x] != x
    p[x] = get(p[x])
  return p[x]
```

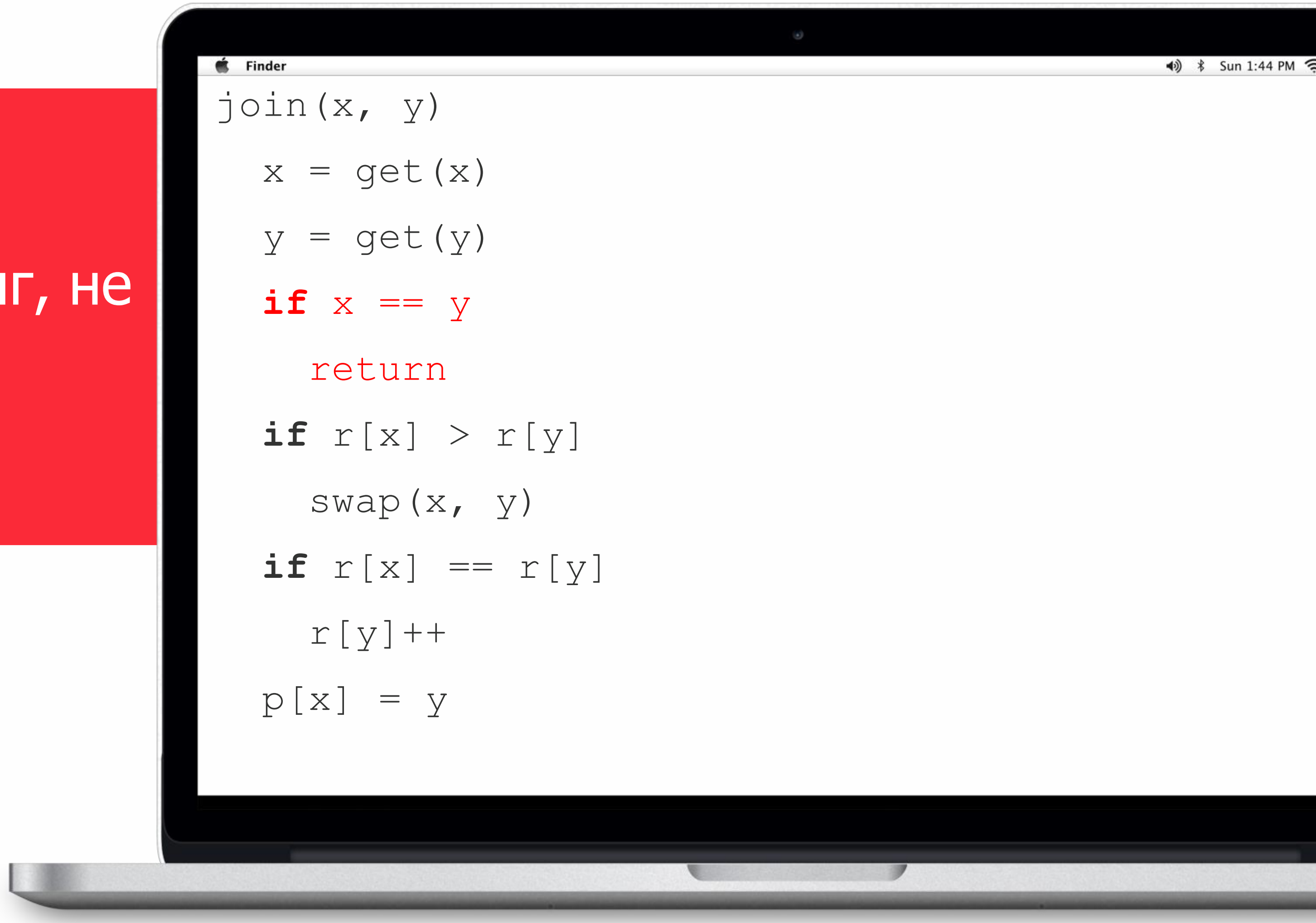
Система непересекающихся множеств

- Применим эвристики
- Кое-что забыли!



Система непересекающихся множеств

- Если этого не сделать будем увеличивать ранг, не меняя множество!



Минимальное остовное дерево

Минимальное остовное дерево

- Дан взвешенный граф
- Остовное дерево – минимальный связный подграф, содержащий все вершины исходного
- Минимальное остовное дерево – остовное дерево, сумма весов ребер, которого минимальна

- Разрез – разбиение V на два непересекающихся множества
- Ребро *пересекает* разрез, если его концы в разных подмножествах

Минимальное остовное дерево

- Лемма о безопасном ребре
- Есть подграф некоторого минимального остовного дерева T′
- Ребро *безопасно*, если при добавлении его в T′, полученный граф также будет являться подграфом некоторого минимального остовного дерева
- Лемма: T′ – подграф некоторого минимального остовного дерева, $< S, T >$ – разрез такой, что ни одно ребро из T′ его не пересекает, ребро $e = (u, v)$ минимальное ребро, пересекающее этот разрез => e – безопасное

Минимальное остовное дерево

- На основе леммы о безопасном ребре есть три наиболее известных алгоритма
  - Алгоритм Крускала/Краскала
  - Алгоритм Прима
  - Алгоритм Борувки

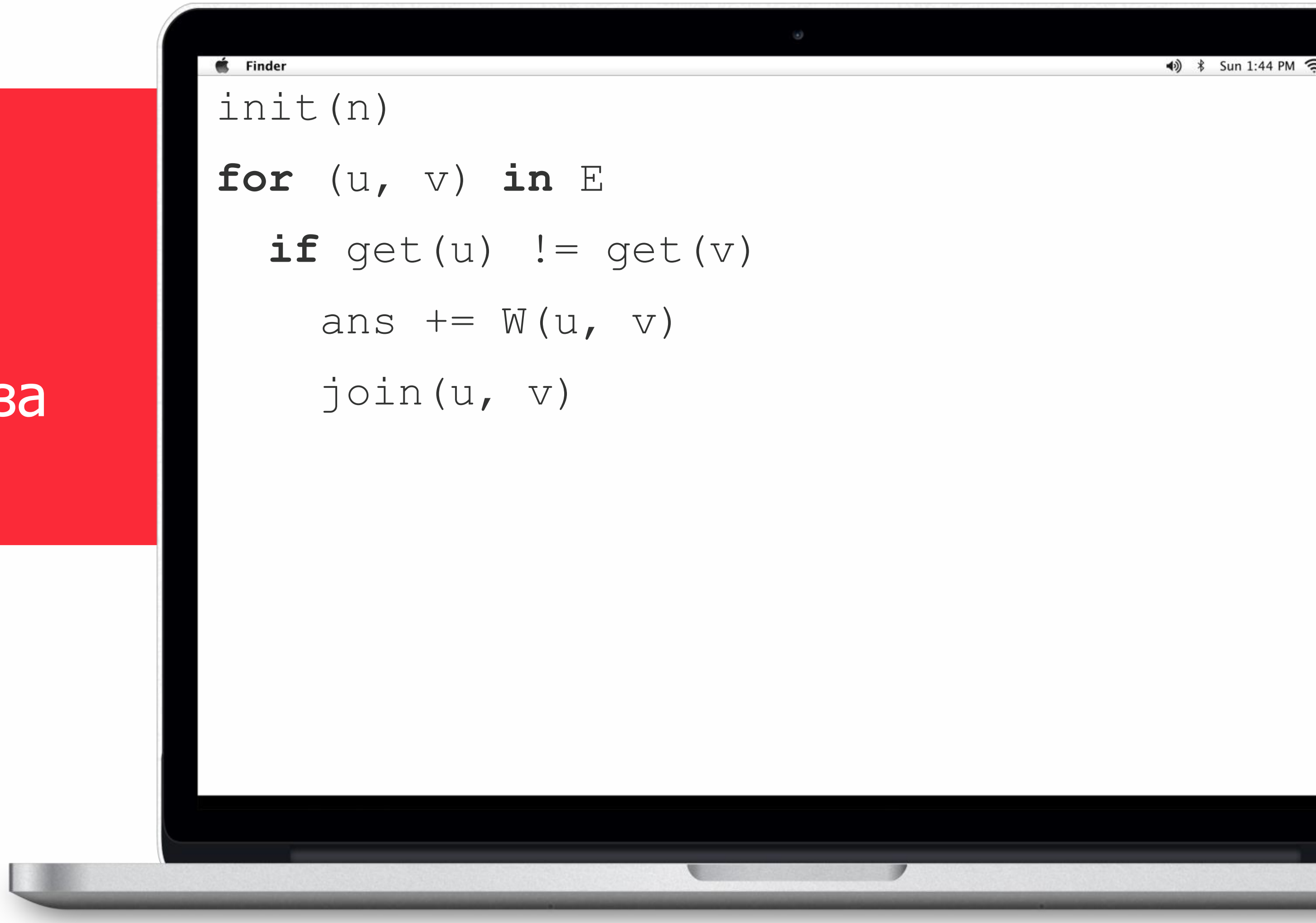
Алгоритм Краскала

Алгоритм Краскала

- По лемме о безопасном ребре, минимальное ребро графа, всегда есть в некотором минимальном остовном дереве
- Как найти следующее ребро?
- Пусть у нас есть некоторый построенный подграф минимального остовного дерева
- Найдем минимальное ребро, которое соединит две разные компоненты связанности (ого, тут как раз разрез)
- Компоненты связанности можем поддерживать СНМ!

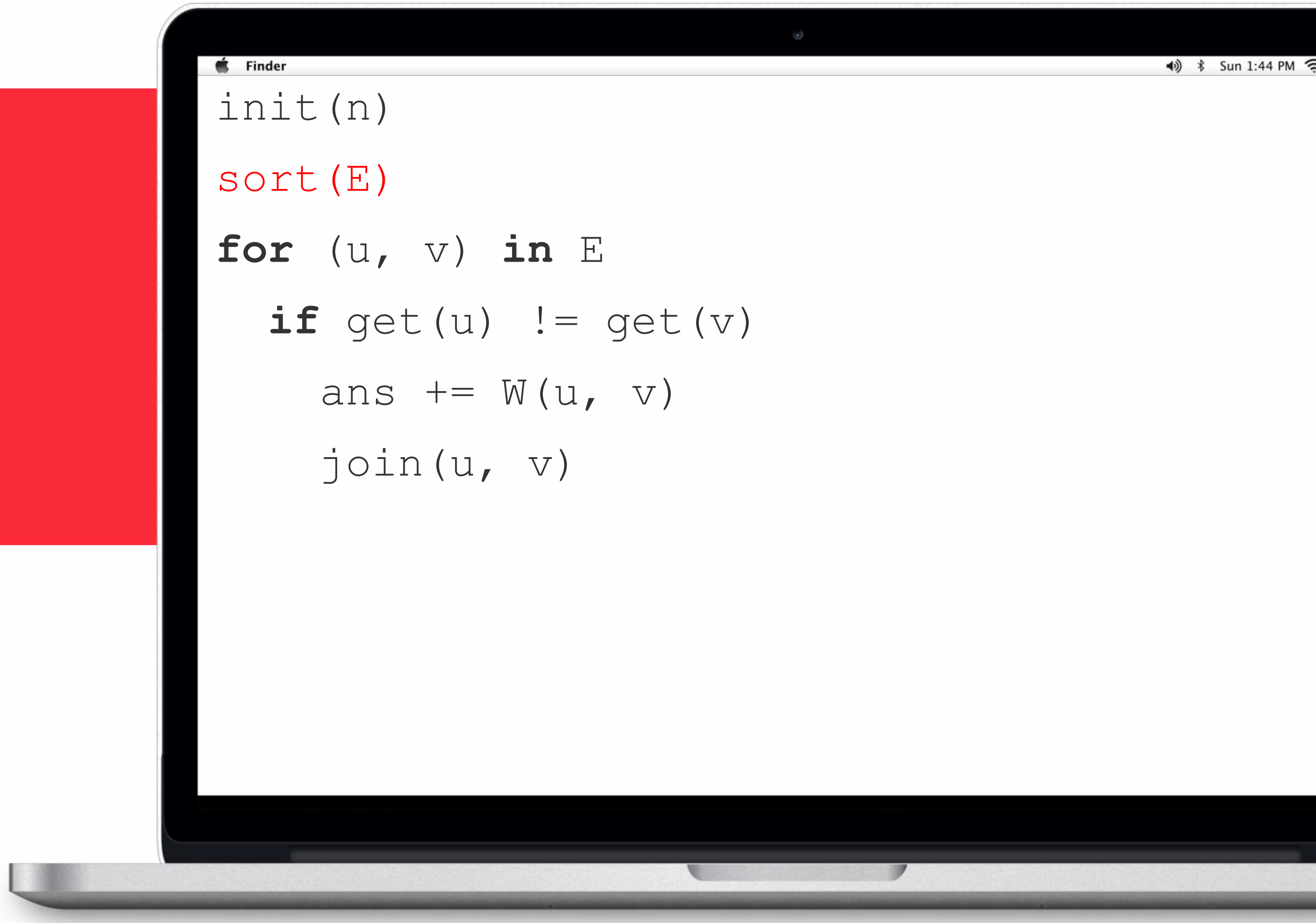
Алгоритм Краскала

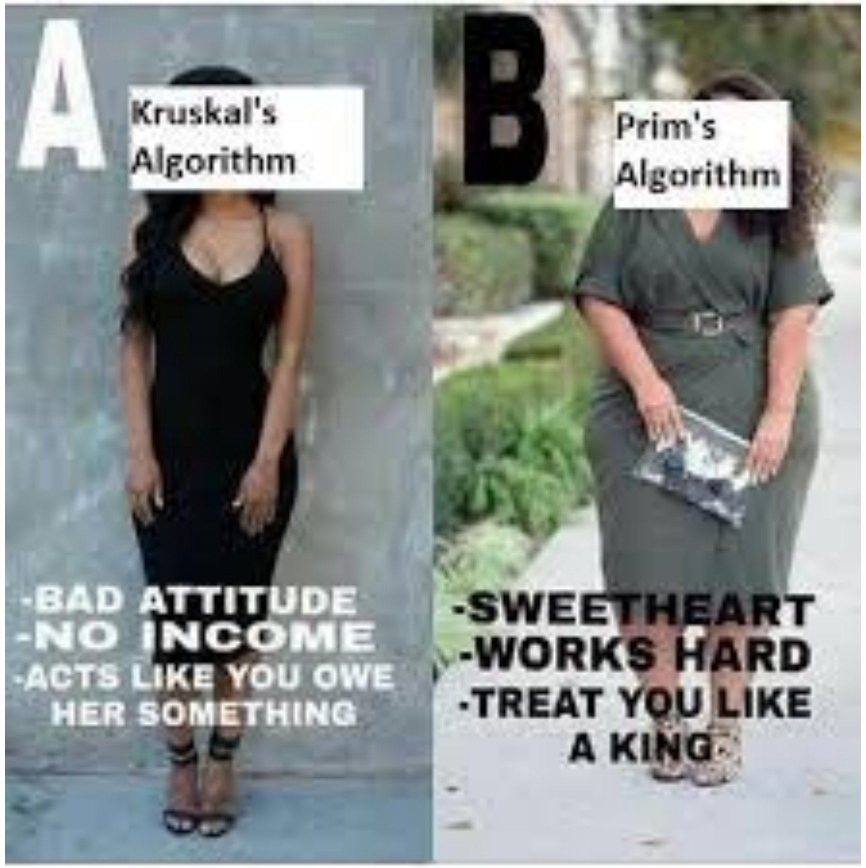
- Сразу к реализации
- Казалось бы работает за O(Eα), но нет!



Алгоритм Краскала

- O(ElogE)!





Алгоритм Прима aka где-то я это уже видел

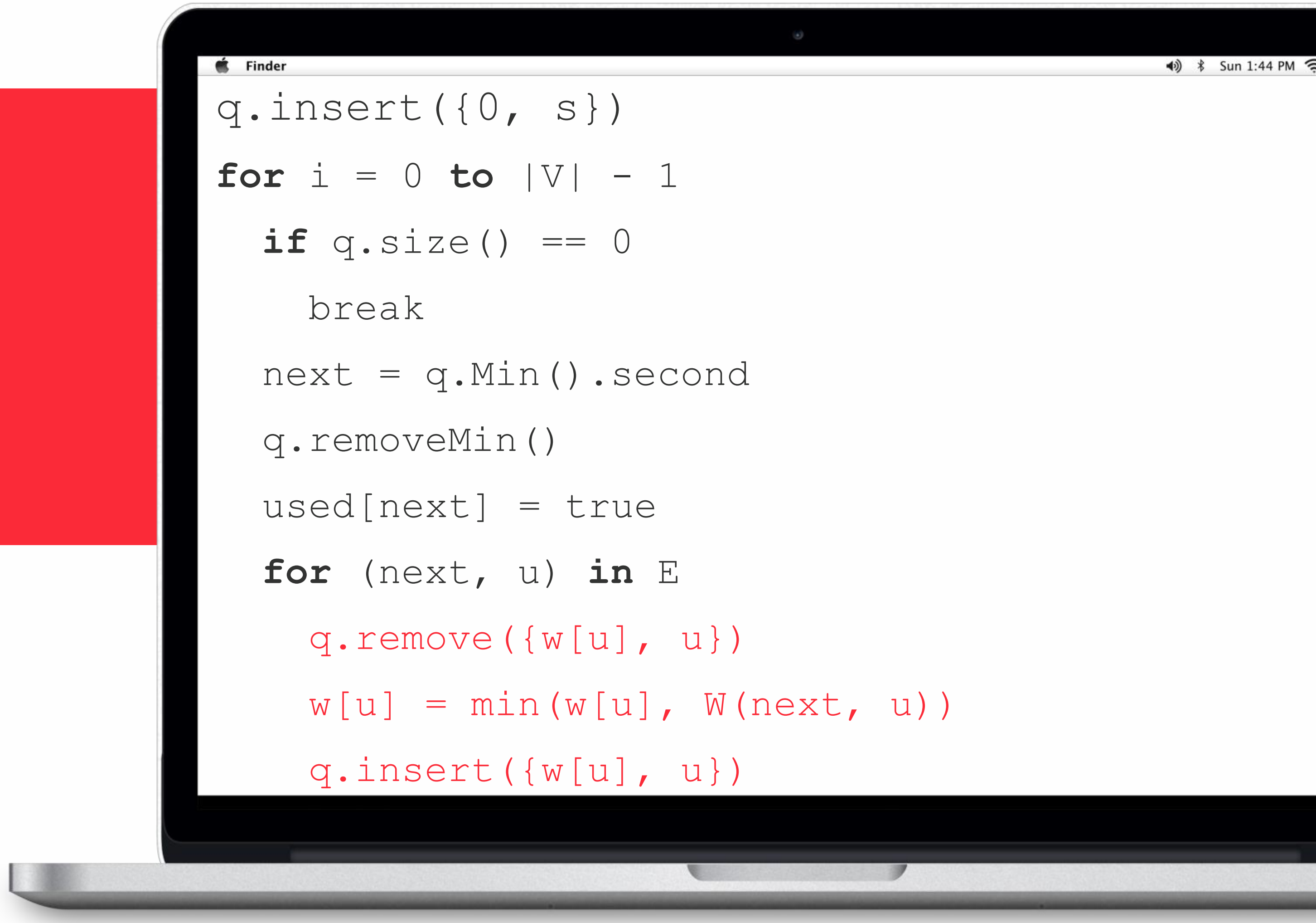
Алгоритм Прима

- Инвариант: есть множество вершин U, для которых уже *построено MST, которое является подмножеством некоторого MST всего графа.*
- Изначально множество U – пустое
- Будем по одной вершине расширять множество U
- Какой переход?
  - Сосед множества U: вершина, которая соединена ребром с вершиной из U, но при этом сама не в U
  - Будем добавлять ~~ближайшего к стартовой вершине~~ соседа множества U*, к которому идет минимальное ребро*
- Будет |V| итераций
- По лемме все будет ок!

Алгоритм Прима

- Даже код из Дейкстры скопипастим!
- Изначально

$$w[s] = 0, w[i] = \infty$$



Алгоритм Прима

- За сколько работает?
- $O(ElogV)$
- Или?
- $O(V^2)$
- То есть на полных графах Прим будет сильно лучше Краскала!

Алгоритм Борувки (десерт)

Алгоритм Борувки

- Три шага к успеху:
    - Изначально каждая вершина графа G — тривиальное дерево, а ребра не принадлежат никакому дереву.
    - Для каждого дерева T найдем минимальное инцидентное ему ребро. Добавим все такие ребра.
    - Повторяем шаг 2 пока в графе не останется только одно дерево T.

Алгоритм Борувки

- Почему это работает?
- Что может пойти не так? (полный граф на три вершины с единичными ребрами)
- Сколько будет итераций?
  - $O(\log(V))$
- Итого:
  - $O(E\log(V))$

Минимальное ориентированное остовное дерево

Минимальное ориентированное остовное дерево

- Дан взвешенный ориентированный граф и стартовая вершина
- Хотим оставить такие ребра, чтобы от стартовой был путь до любой другой
- Суммарный вес: минимальный
- Логично, что все вершины достижимы из стартовой!

Алгоритм двух китайцев (Чу и Лю)

- Заметим, что если взять все ребра входящие в вершину v и прибавить к весу каждого x, вес искомого MST измениться ровно на x
- Тогда для каждой вершины найдем минимальное входящее ребро и вычтем его вес из остальных входящих
- Рассмотрим граф, построенный на нулевых ребрах
- Либо он дерево и все уже ок
- Либо он состоит из компонент связности с одним циклом и деревьев

Алгоритм двух китайцев (Чу и Лю)

- Либо он состоит из компонент связности с одним циклом и деревьев
- Сделаем конденсацию, повторим все сначала!
- Как найти остовное дерево в нас, если знаем остовное дерево в конденсации?
- Найдем ребро, которое входит в компоненту сильной связности
- Вес ребер в компоненте ноль!
- Запустим из нее дфс или удалим второе входящее ребро
- Профит!

Алгоритм двух китайцев (Чу и Лю)

- За сколько работает?
- $O(V)$ итераций
- Итого: $O(VE)$

Все!