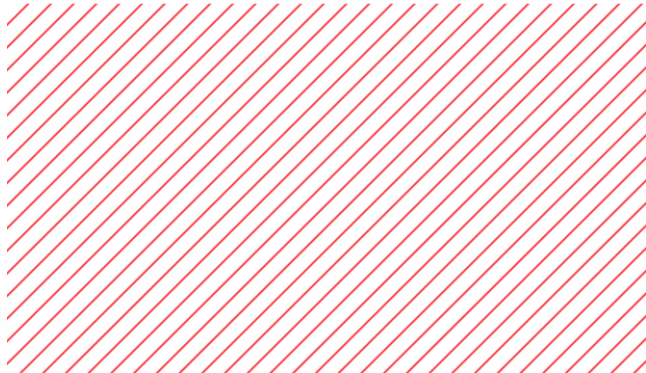
академия больших данных

mail.ru group

Деревья поиска 2

Шовкопляс Григорий

Введение в алгоритмы и структуры данных

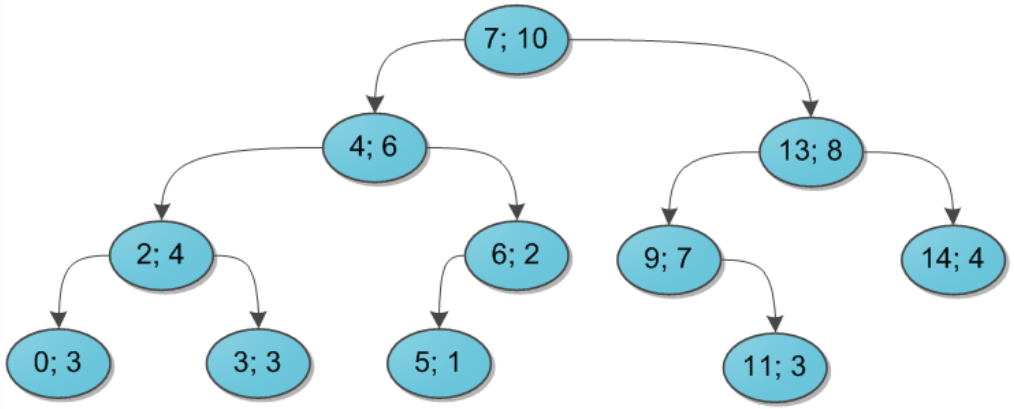


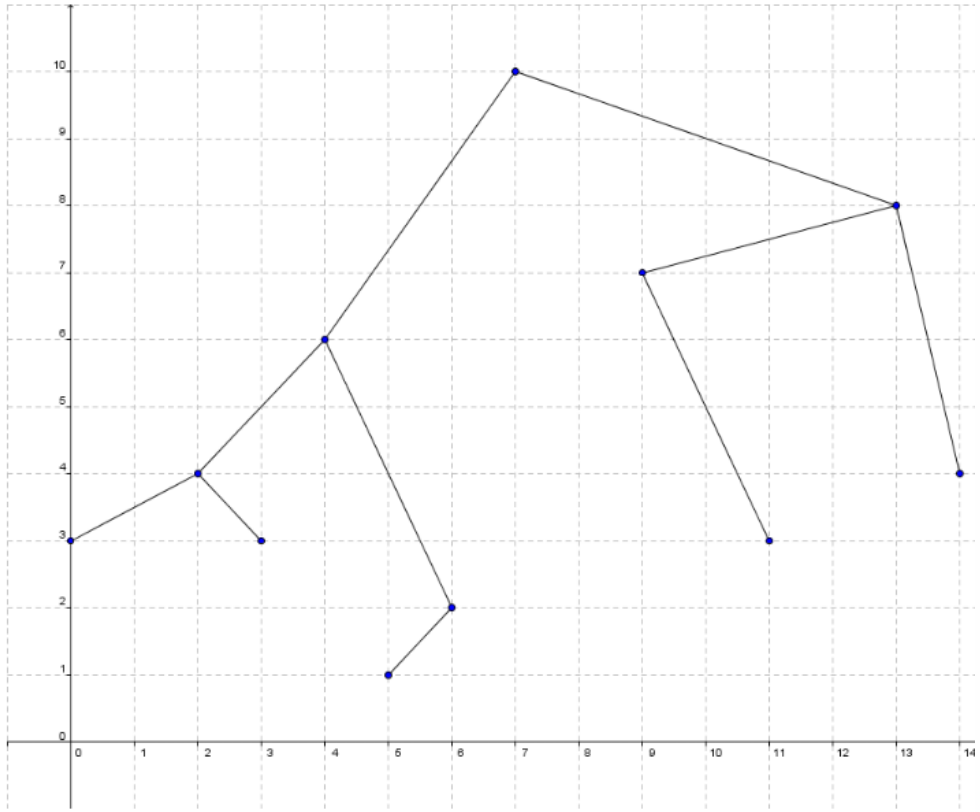
Декартово дерево

Декартово дерево

- Объединим кучу (приоритетную очередь) и дерево поиска
- Храним в вершине два значения: (X, Y)
  - X – ключ
  - Y – приоритет
- Что если сделать структуру данных, что по x дерево поиска, а по y куча?

Декартово дерево





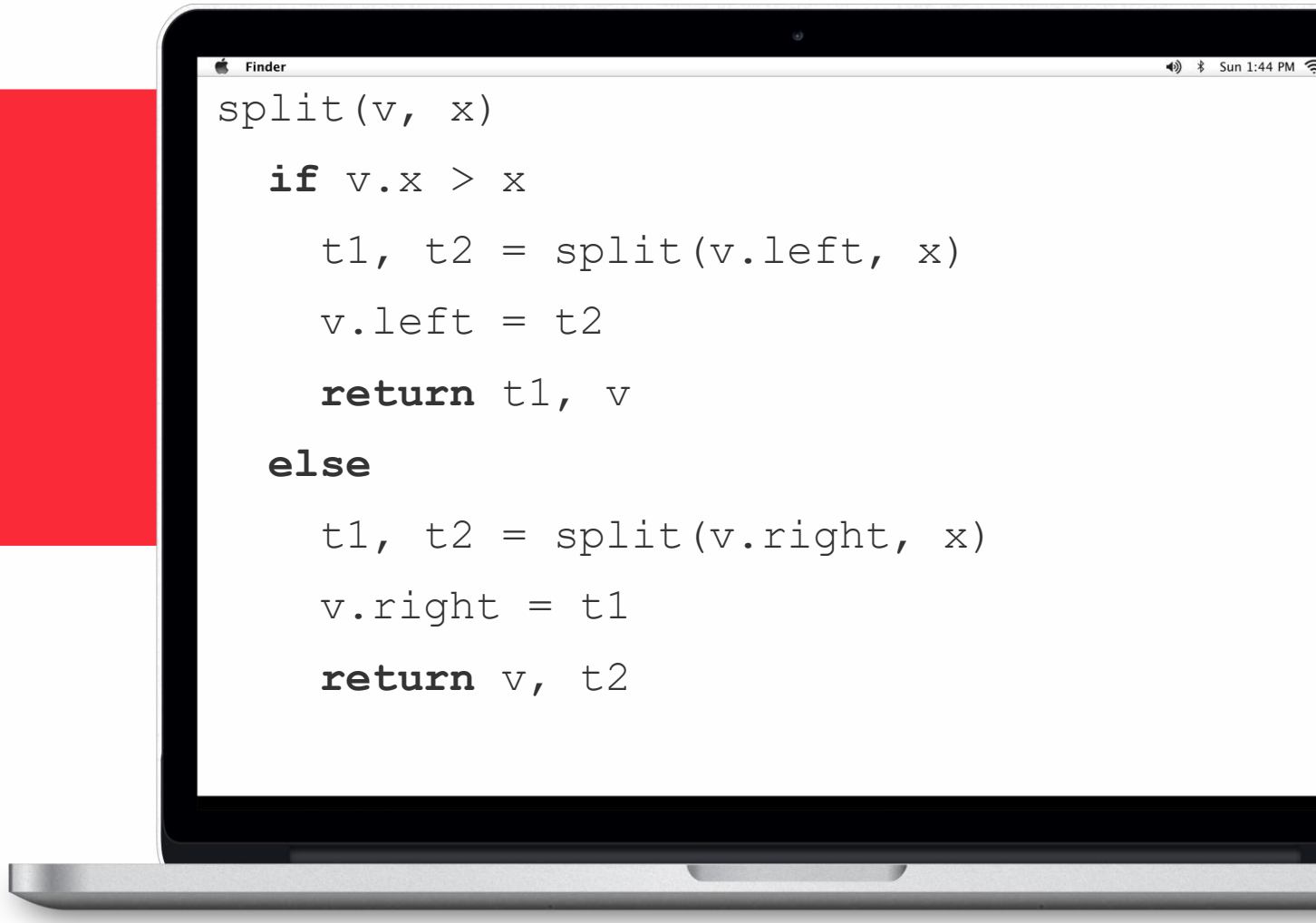
Декартово дерево

- Поиск, как в обычном дереве поиска (логично)
- Split(T, x)
  - Разделяет дерево T на два по ключу x
  - Возвращает два дерева T_1, T_2
  - T_1 – все ключи $\leq x$, T_2 – все ключи $> x$
- Merge(T_1, T_2)
  - Если все ключи T_1 меньше всех ключей T_2
  - Вернет дерево T, их объединение
- Операции модификации можно выразить через Split и Merge

Декартово дерево

Split(v, x):

- v.x > x
  - v и v.right точно в T_2
- Иначе
  - v и v.left точно в T_1



Декартово дерево

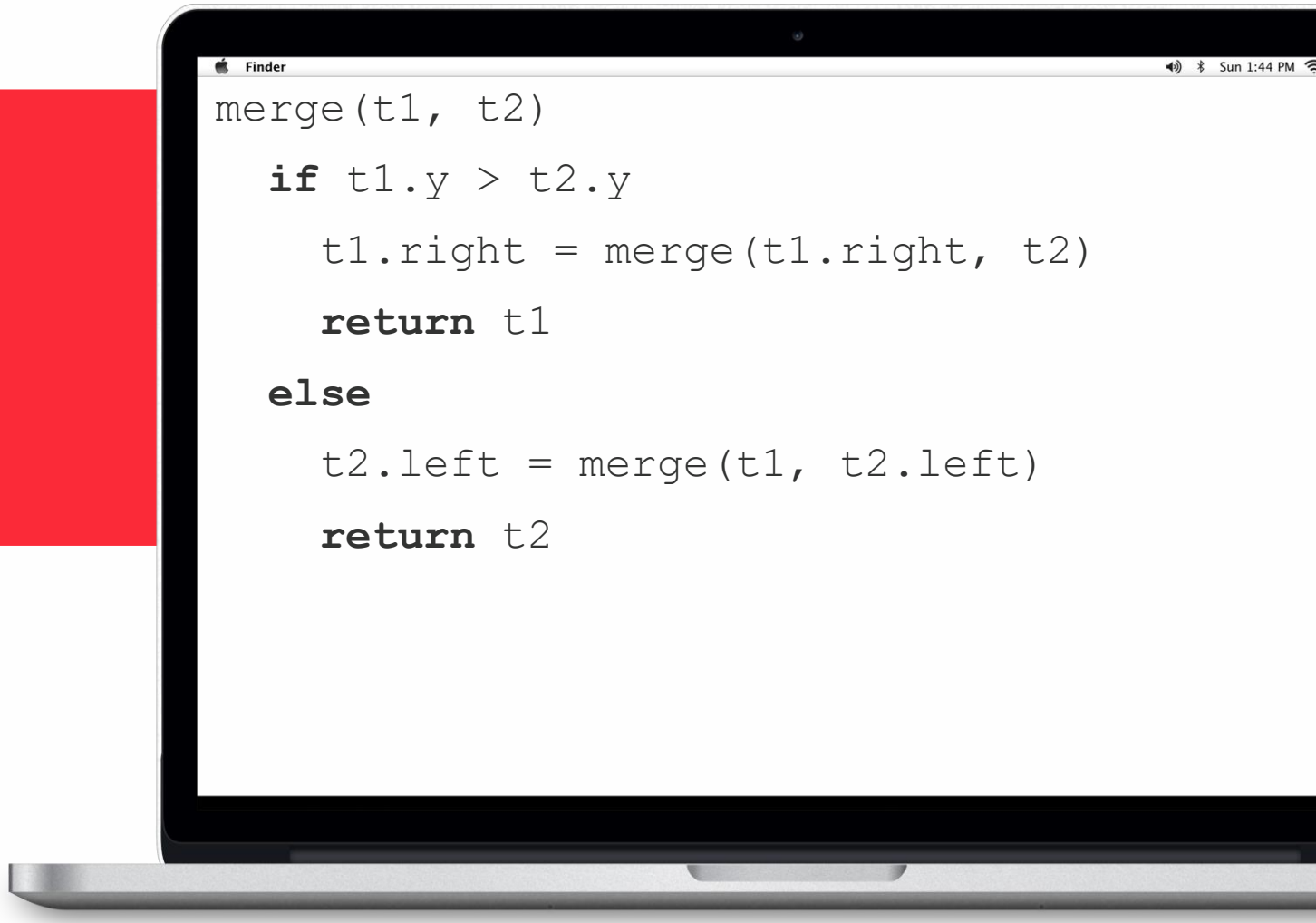
Терминальное условие!

Время работы O(h)

Декартово дерево

Merge(t1, t2):

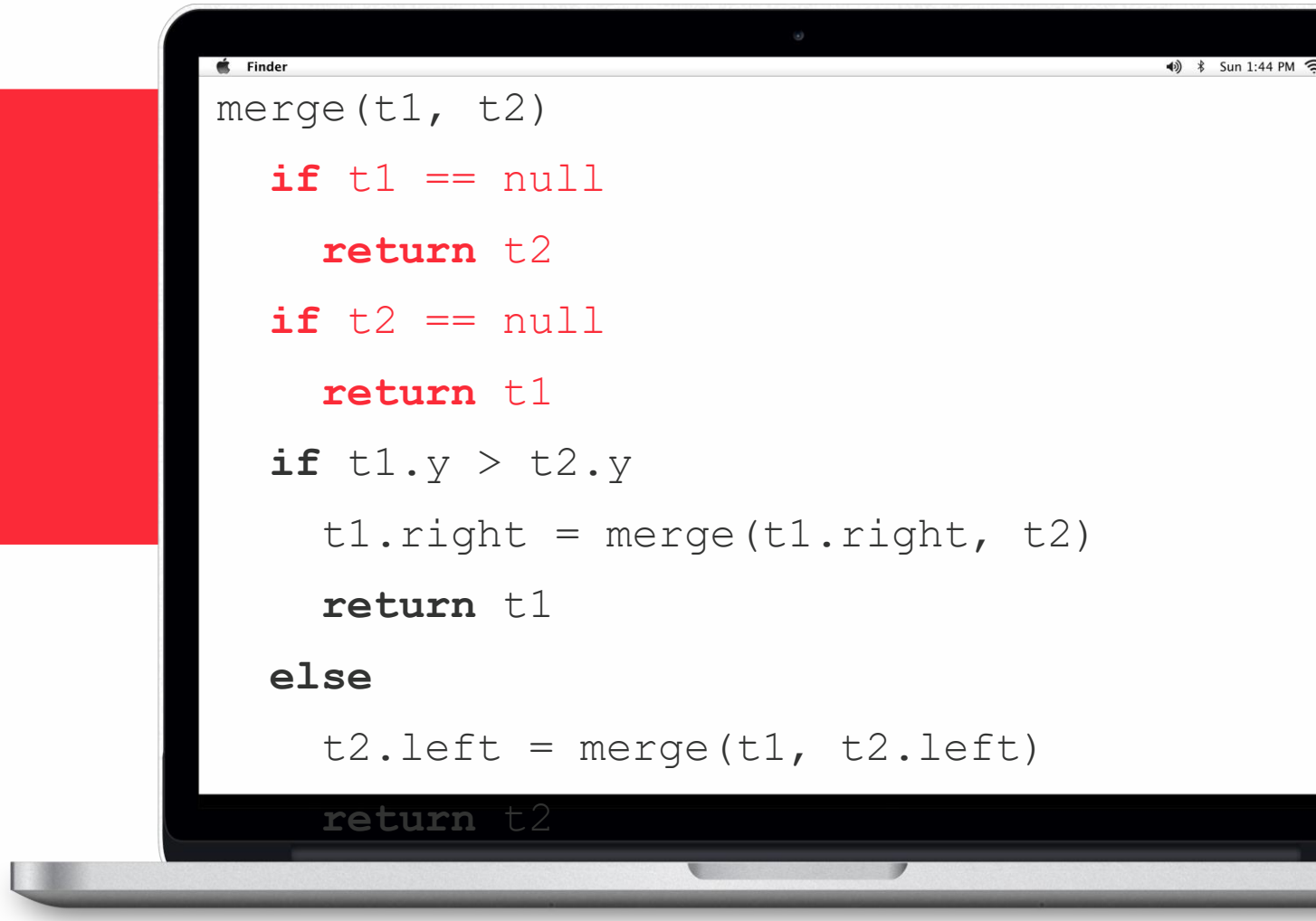
- t1.y > t2.y
  - t1 - корень
- Иначе
  - t2 - корень



Декартово дерево

Хмм... чего-то не хватает

Время работы O(h)



Декартово дерево

- Insert(t, x, y)
  - $Split(t, x) \rightarrow Merge(t_1, new) \rightarrow Merge(t_1, t_2)$
- Реализация без merge
  - Найдем первую вершину на пути поиска, что v.y < y
  - Split(v, x)
  - new.left = t1, new.right = t2, v = new
  - O(h) вместо O(3h)

Декартово дерево

- remove(t, x)
  - $Split(t, x) \rightarrow Split(t_1, x-1) \rightarrow Merge(t_{11}, t_2)$
- Реализация без split
  - Найдем нужную вершину
  - v = Merge(v.left, v.right)
  - Не забудем почистить память
  - O(h) вместо O(3h)

Время доказательств

- Лемма: для множества конкретных пар, декартово дерево строится единственным образом
- А где сбалансированность?
- Ключи нужны, приоритеты не очень…
- Теорема: если сделать ключи случайными, средняя глубина $O(\log n)$

Время доказательств

- Теорема: если сделать ключи случайными, средняя глубина $O(\log n)$
- Как и ранее считаем, что все приоритеты различны
- Пусть v_k – вершина с k-м по величине ключом
- $A_{i,j}$ – 1, если v_i предок v_j, 0 иначе
- $d(v)$ глубина вершины v
- Тогда $d(v_k) = \sum_{i=1}^{n} A_{i,k}$
- $E(d(v_k)) = \sum_{i=1}^{n} P(A_{i,k} = 1)$
- Осталось оценить вероятность $P(A_{i,k} = 1)$

Время доказательств

- Лемма: для любых $i \neq k$ v_i предок v_k имеет наибольший приоритет из $\{v_i, v_{i+1} \dots v_k\}$
- Четыре случая:
  - v_i – корень
  - v_k – корень
  - v_{other} – корень, v_i и v_k в разных поддеревьях
  - v_{other} – корень, v_i и v_k в одном поддереве

Время доказательств

- Что там с $P\left(A_{i,k}=1\right)$?
- $P\left(A_{i,k}=1\right)=0;\ k=i$
- $P\left(A_{i,k}=1\right)=\frac{1}{k-i+1};k>i$
- $P\left(A_{i,k}=1\right)=\frac{1}{i-k+1};k<i$
- $E\left(d(v_k)\right)=\sum_{i=1}^{k-1}\frac{1}{k-i+1}+\sum_{i=k+1}^{n}\frac{1}{i-k+1}$
- $\sum_{i=1}^{n}\frac{1}{i}\leq \ln(n)+1$
- $E\left(d(v_k)\right)\leq \ln(k)+\ln(n-k)+2=O(\log(n))$

Время выводов

- $Insert(t, x, y) \rightarrow \mathrm{Insert(t, x, rand())}$
- Получаем дерево поиска со средней высотой O(log(n))

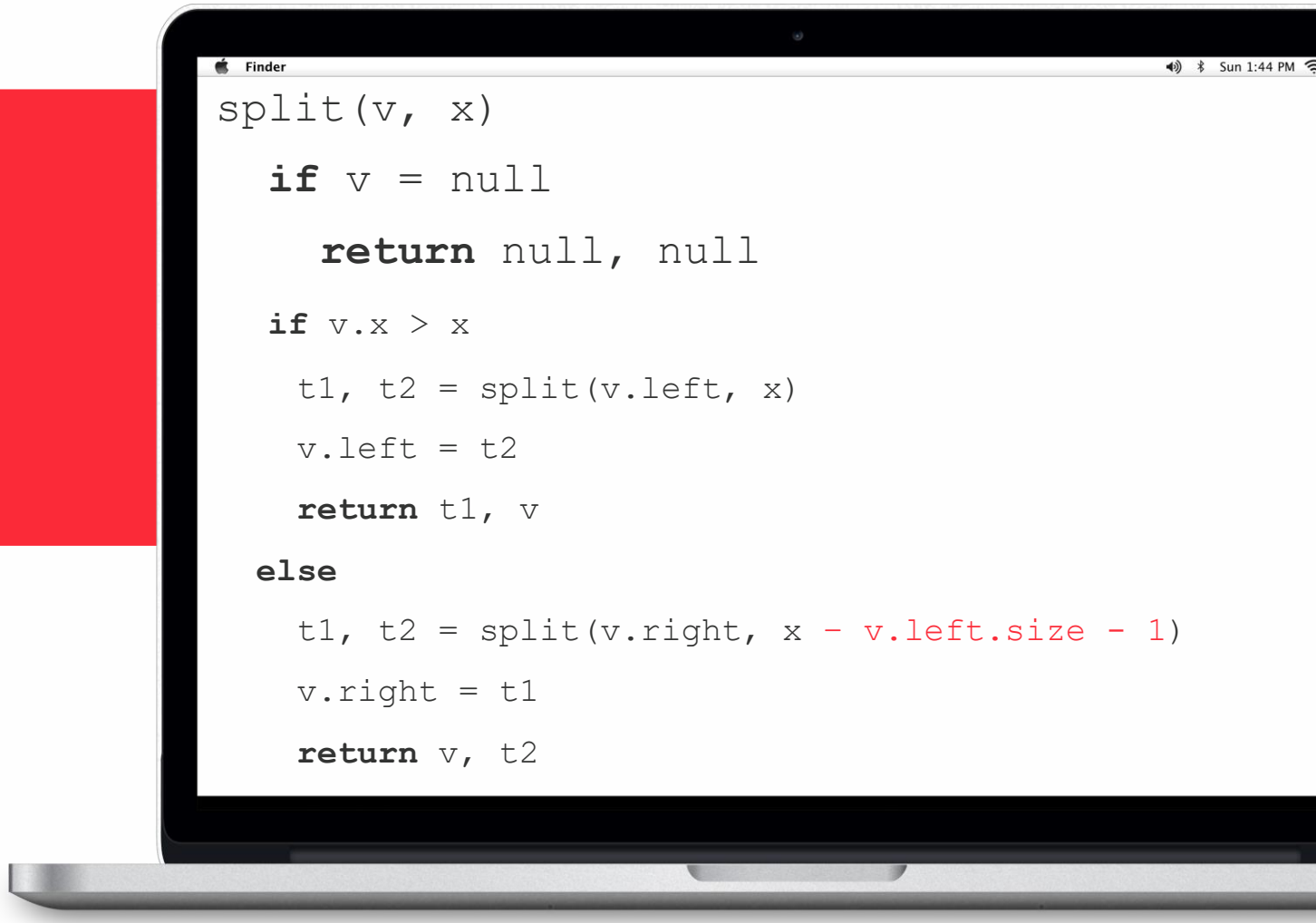
Неявный ключ

Неявный ключ

- Что будет, если хранить в вершине три значения:
  - Ключ, значение, случайный приоритет
  - И вставить в декартово дерево массив
  - $Insert(i, a_i, rand())$
- Если дерево содержит все ключи от 1 до n, ключи можно не хранить, а вычислять «на ходу»
- Для этого нужно хранить размеры поддеревьев

Декартово дерево

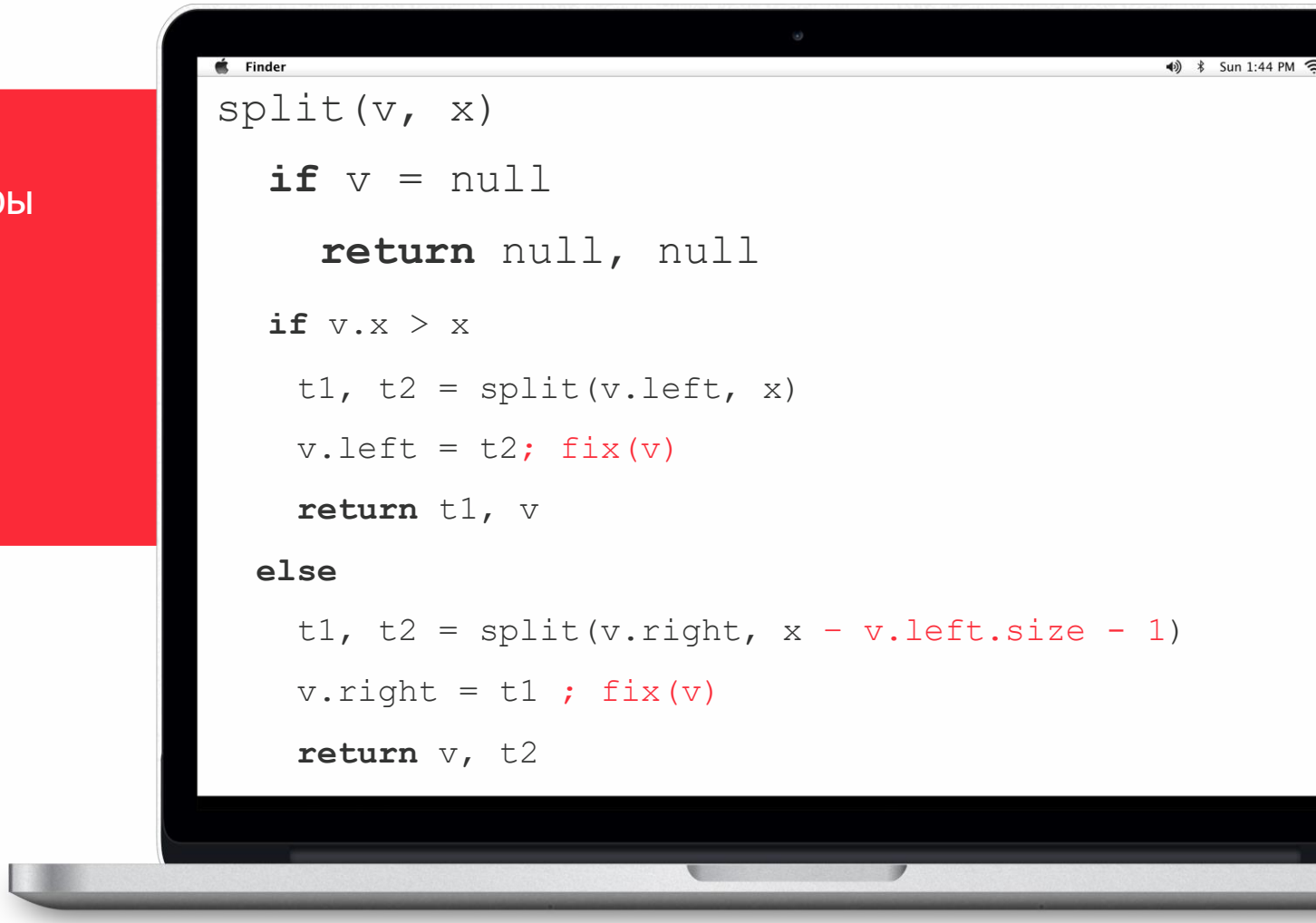
Как измениться split?



Декартово дерево

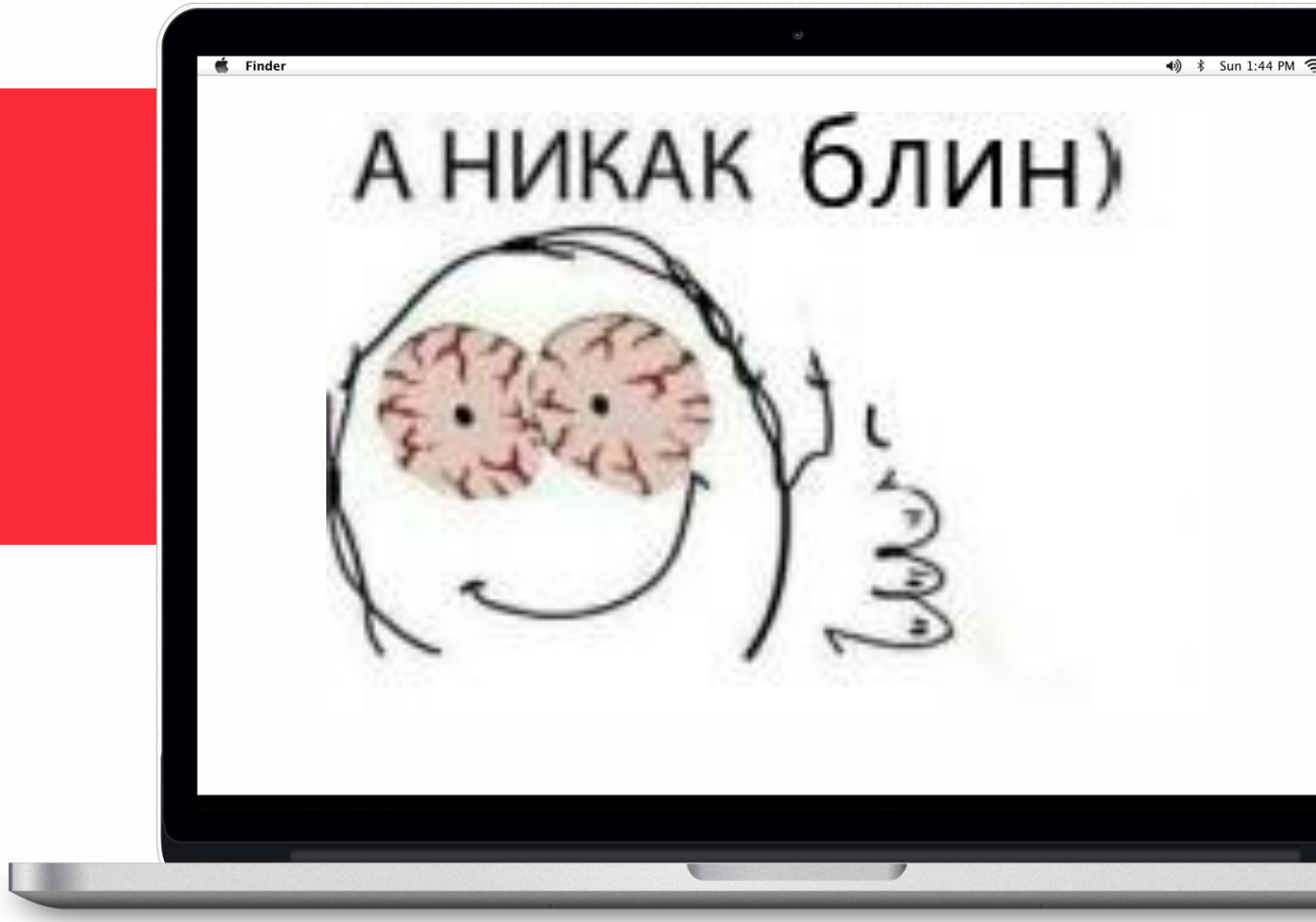
Надо не забыть пересчитать размеры поддерервьев!

```
fix(v):
  v.size = getSize(v.left)
           + getSize(v.right) + 1
```



Декартово дерево

Как измениться merge?



Неявный ключ

- Что нам это дает?
- Массив, в котором мы можем вставлять и удалять в любое место за O(log(n))
- Что можем еще?
- Все то же самое, что и дерево отрезков + п.1
- Можем что-нибудь еще?
- Да, реверсить подотрезки массива!

Все!