Алгоритмы и структуры данных

Кучи II

CS Center, Новосибирск

Преобразование массив -> куча

Кучу из N элементов можно построить за O(N logN)

Преобразование массив -> куча

Кучу из N элементов можно построить за O(N logN)

- Можно ли быстрее?

Преобразование массив -> куча

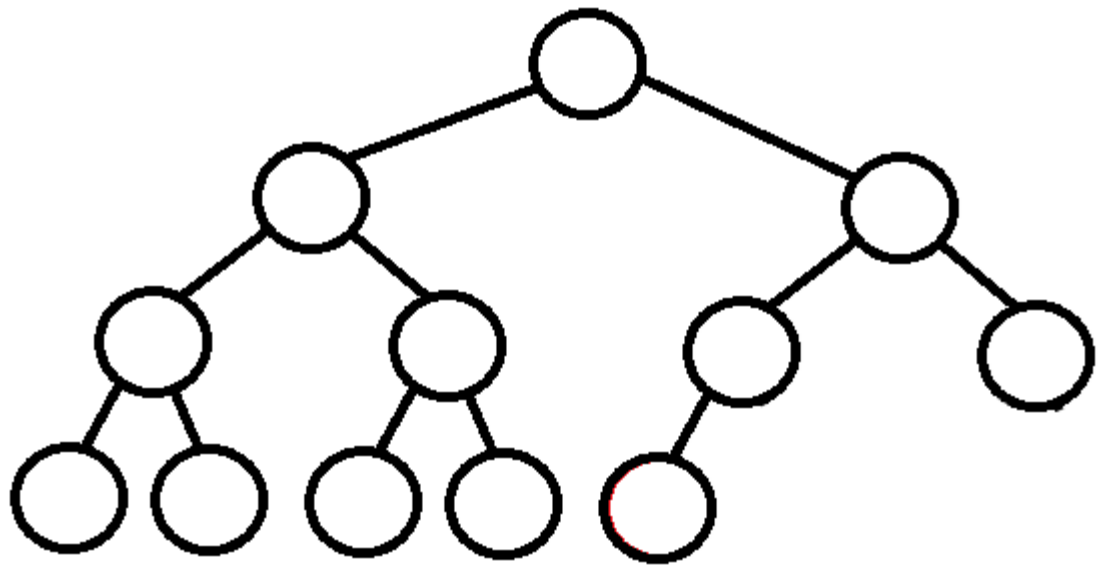
Кучу из N элементов можно построить за O(N logN)

- Можно ли быстрее?

  Научимся за O(N)

Преобразование массив -> куча за O(N)

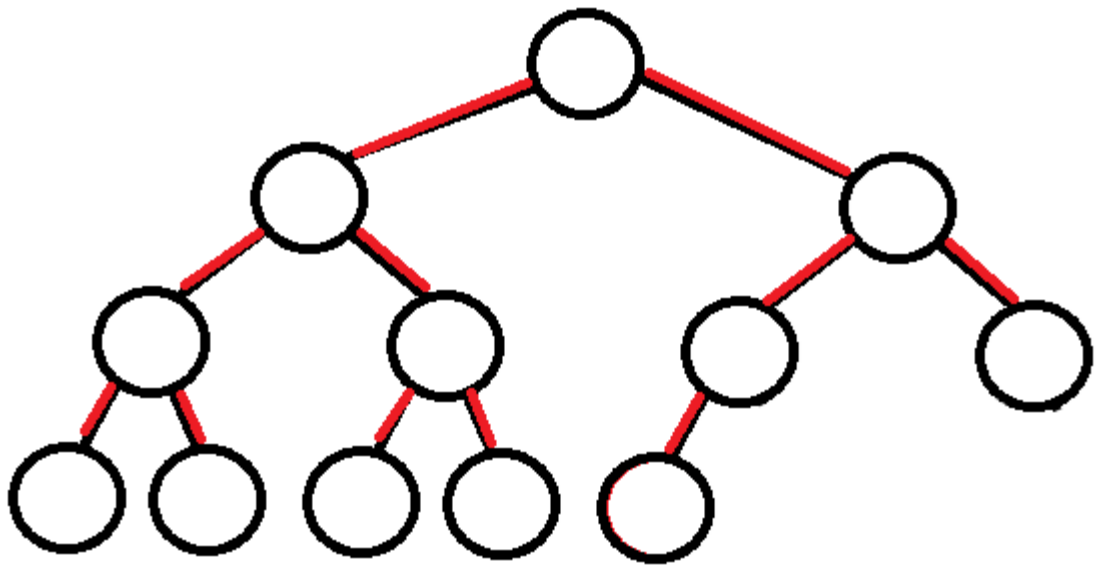
Рассмотрим дерево на массиве



Преобразование массив -> куча за O(N)

Рассмотрим дерево на массиве

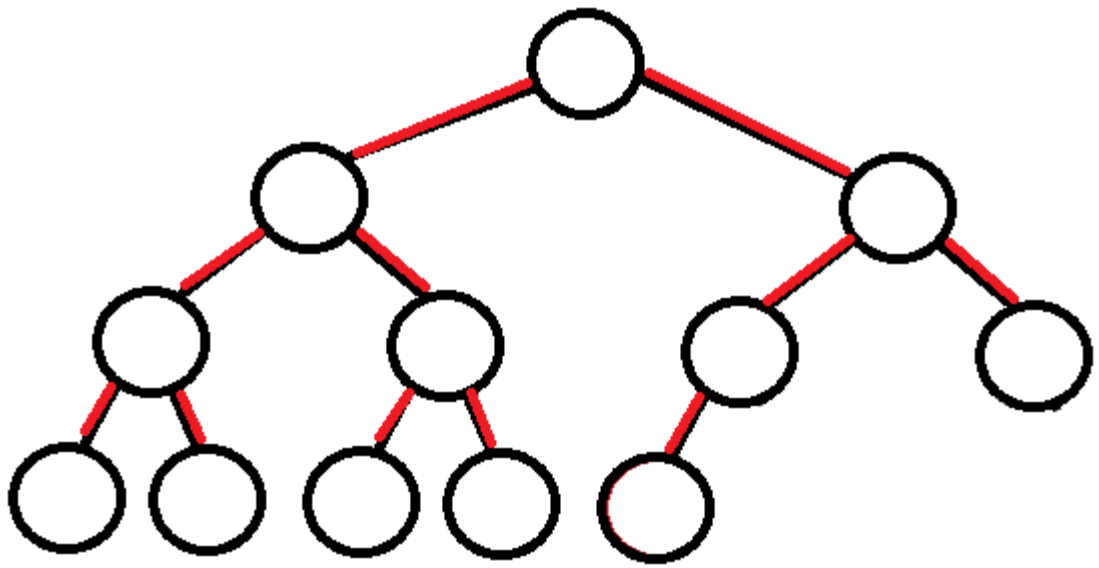
- Изначально в нём могут конликтовать все рёбра



Преобразование массив -> куча за O(N)

Рассмотрим дерево на массиве

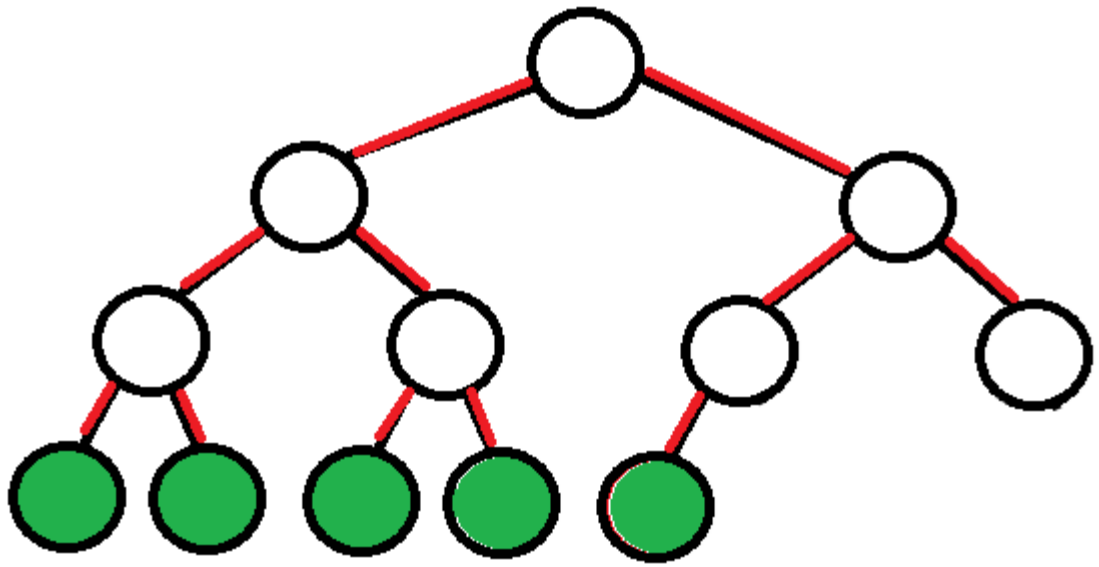
- Изначально в нём могут конликтовать все рёбра
- Будем чинить поддеревья в порядке от листьев к корню



Преобразование массив -> куча за O(N)

Рассмотрим дерево на массиве

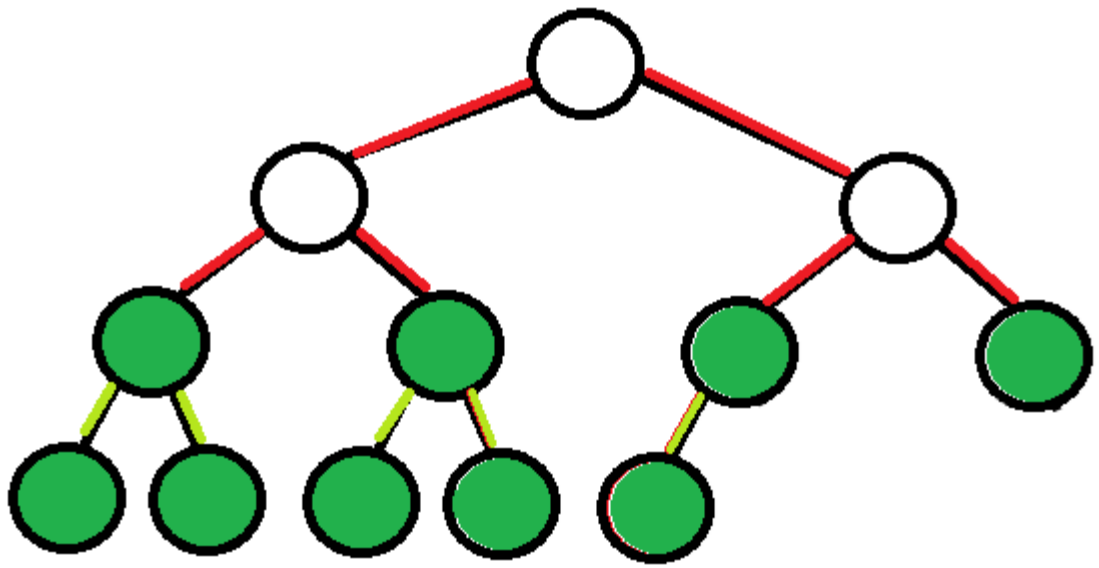
- Изначально в нём могут конликтовать все рёбра
- Будем чинить поддеревья в порядке от листьев к корню
  - В нижнем слое всё ok



Преобразование массив -> куча за O(N)

Рассмотрим дерево на массиве

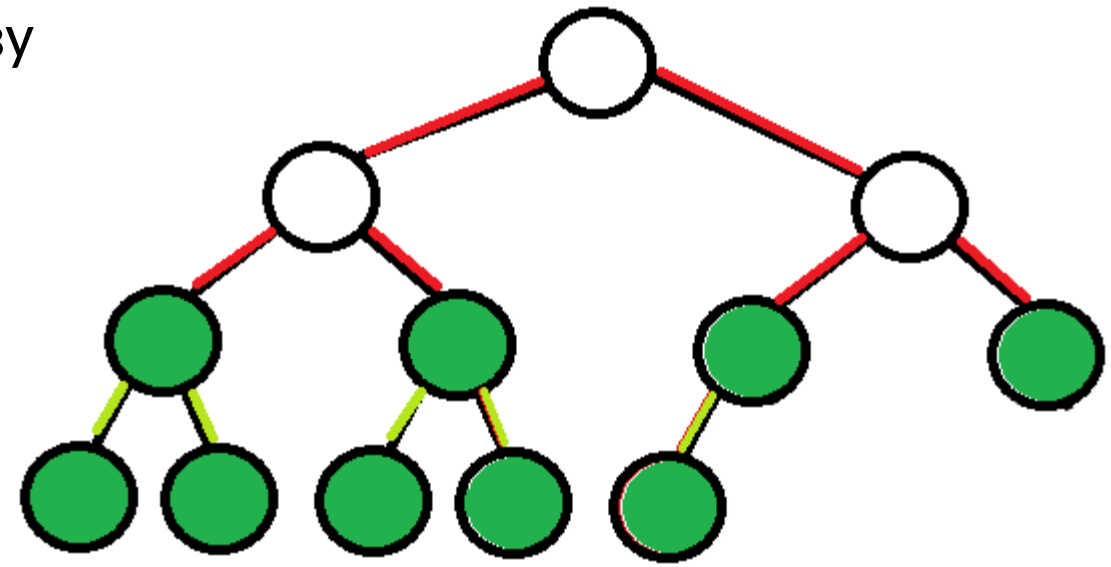
- Изначально в нём могут конликтовать все рёбра
- Будем чинить поддеревья в порядке от листьев к корню
  - В нижнем слое всё ok



Преобразование массив -> куча за O(N)

Рассмотрим дерево на массиве

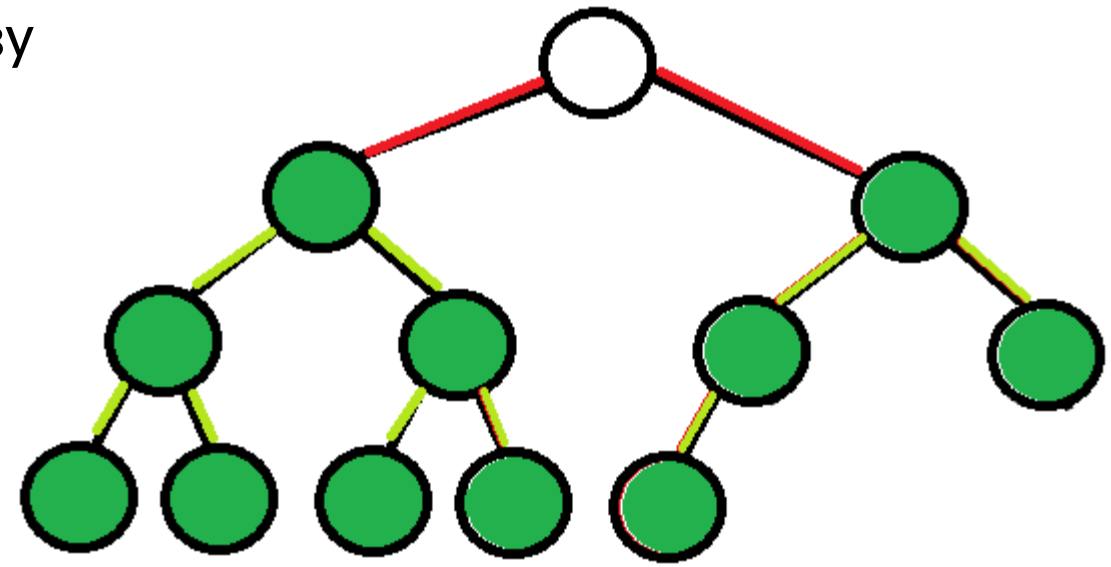
- Изначально в нём могут конликтовать все рёбра
- Будем чинить поддеревья в порядке от листьев к корню
  - В нижнем слое всё ok
  - Сделаем SiftDown во втором слое снизу



Преобразование массив -> куча за O(N)

Рассмотрим дерево на массиве

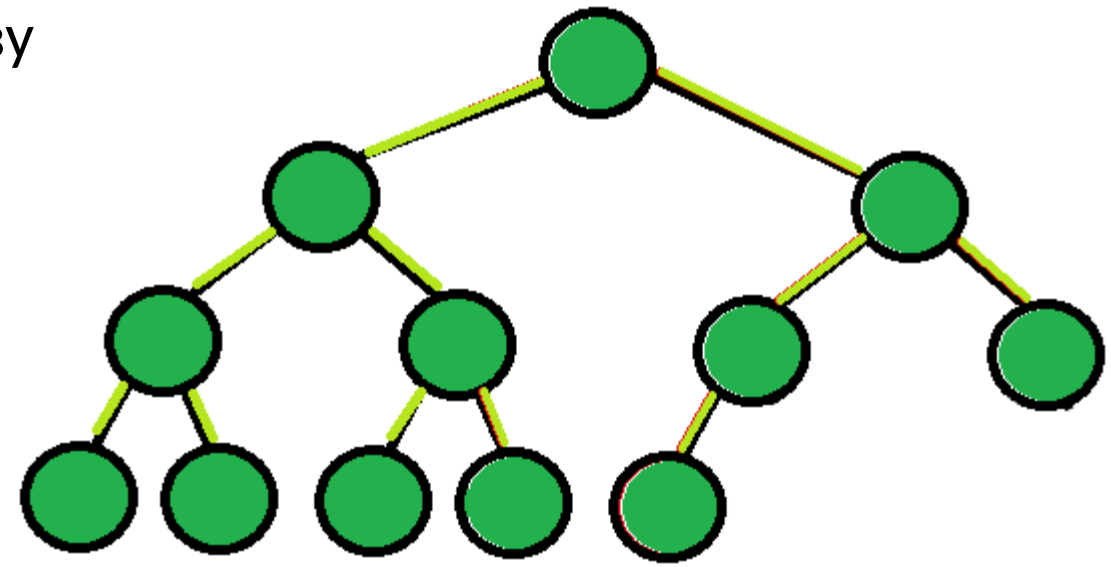
- Изначально в нём могут конликтовать все рёбра
- Будем чинить поддеревья в порядке от листьев к корню
  - В нижнем слое всё ok
  - Сделаем SiftDown во втором слое снизу
  - Поднимаемся выше



Преобразование массив -> куча за O(N)

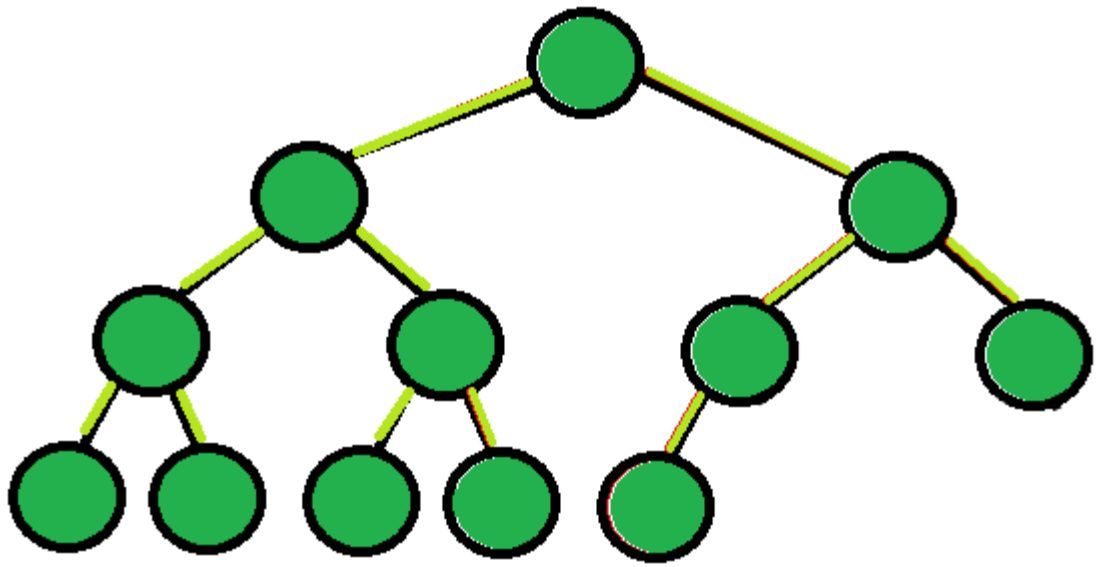
Рассмотрим дерево на массиве

- Изначально в нём могут конликтовать все рёбра
- Будем чинить поддеревья в порядке от листьев к корню
  - В нижнем слое всё ok
  - Сделаем SiftDown во втором слое снизу
  - Поднимаемся выше
  - …
  - Готово!



Преобразование массив -> куча за O(N)

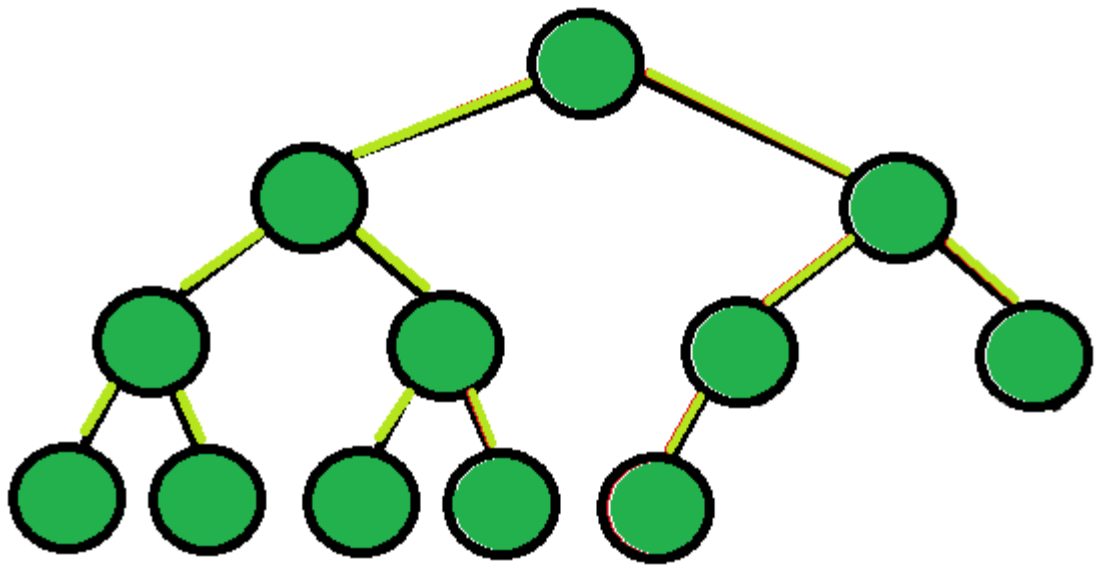
Будем чинить поддеревья в порядке от листьев к корню: SiftDown



Преобразование массив -> куча за O(N)

Будем чинить поддеревья в порядке от листьев к корню: SiftDown

T(0) = 0

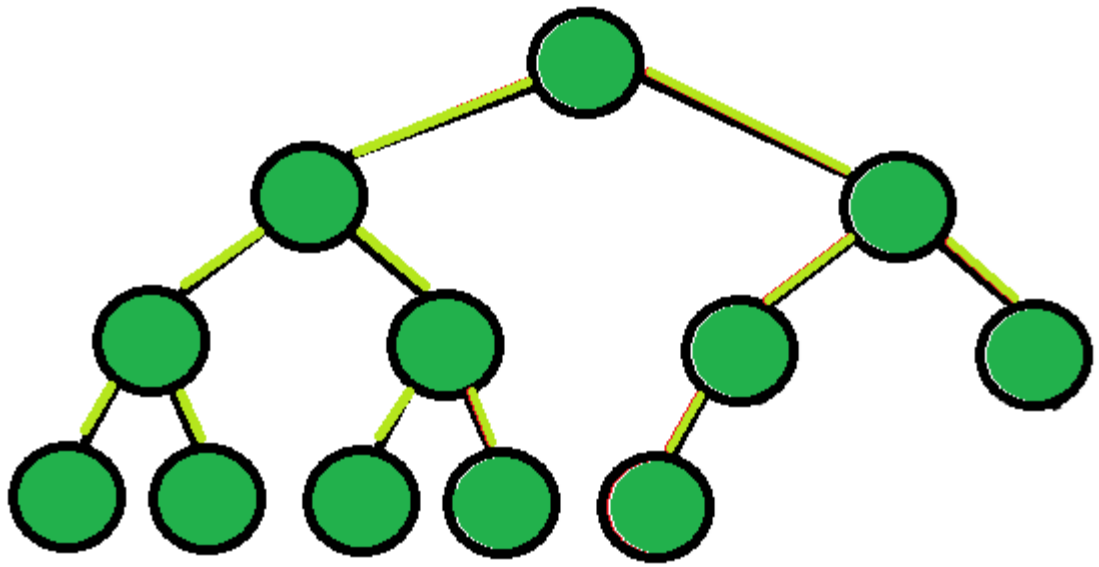


Преобразование массив -> куча за O(N)

Будем чинить поддеревья в порядке от листьев к корню: SiftDown

$T(0) = 0$

$T(k+1) = (k + 1) + 2 * T(k)$



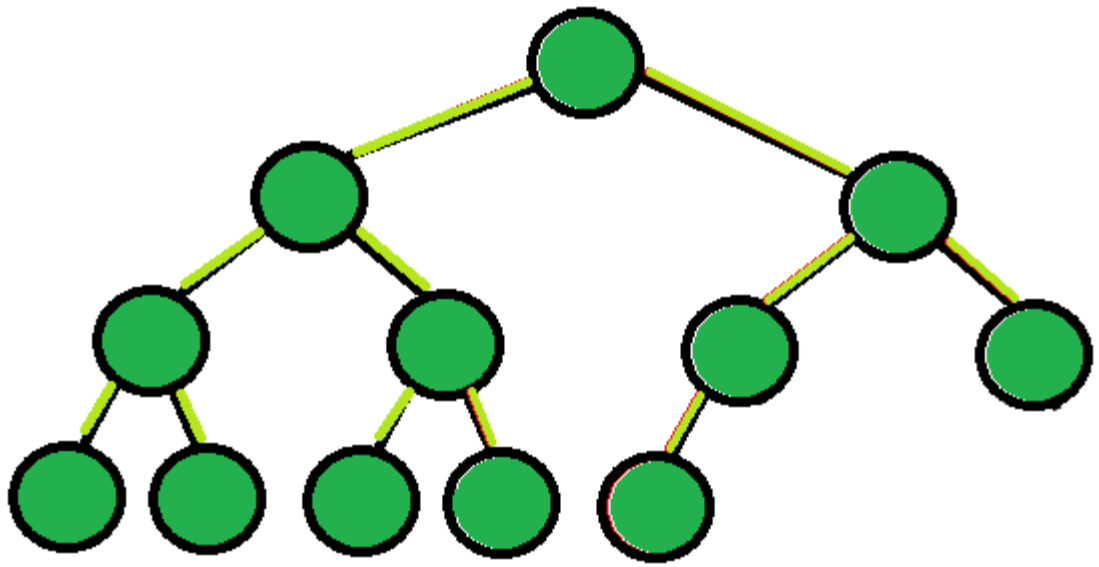
Преобразование массив -> куча за O(N)

Будем чинить поддеревья в порядке от листьев к корню: SiftDown

$T(0) = 0$

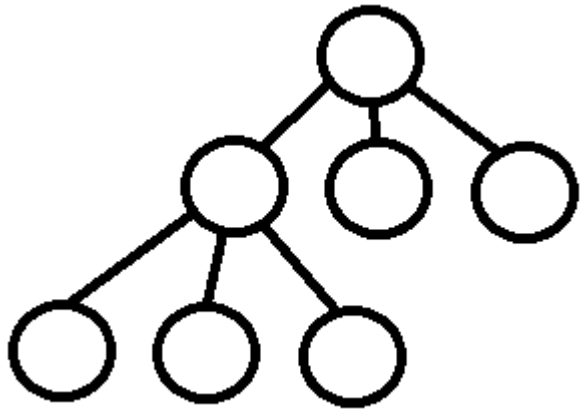
$T(k+1) = (k + 1) + 2 * T(k)$

$k = \log(N)$, $T = O(N)$



k-ичная куча

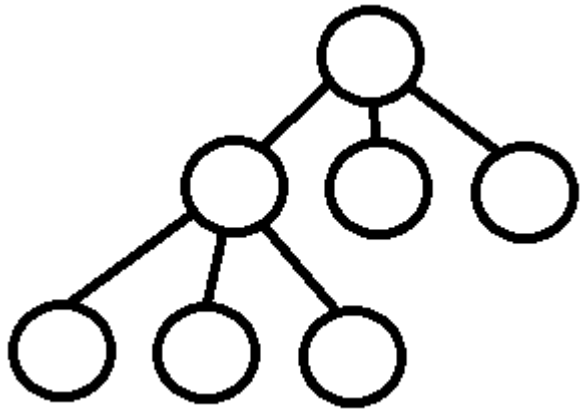
Идея: изменить число детей у вершин дерева



k-ичная куча

Идея: изменить число детей у вершин дерева

- Бинарная куча – частный случай, k=2

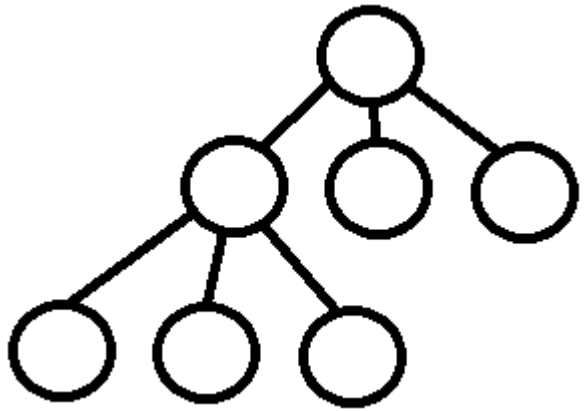


k-ичная куча

Идея: изменить число детей у вершин дерева

- Бинарная куча – частный случай, k=2

Что нам даст болшее k?



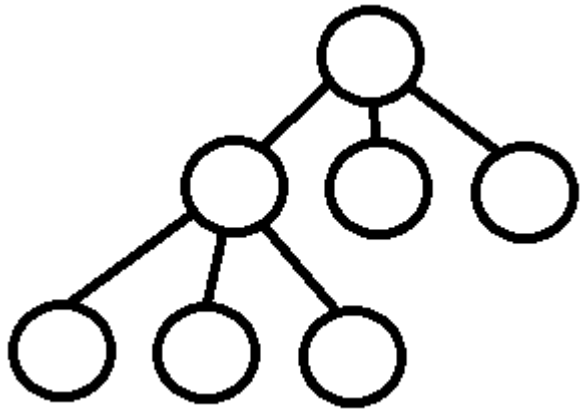
k-ичная куча

Идея: изменить число детей у вершин дерева

- Бинарная куча – частный случай, k=2

Что нам даст болшее k?

- Высота дерева будет меньше



k-ичная куча

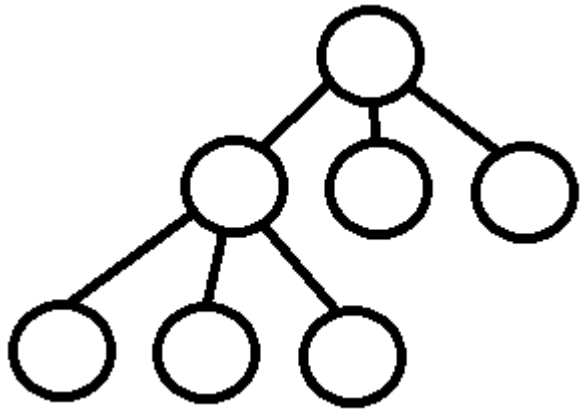
Идея: изменить число детей у вершин дерева

- Бинарная куча – частный случай, k=2

Что нам даст болшее k?

- Высота дерева будет меньше

  $T_{SiftUp} = O(\log_k N)$



k-ичная куча

Идея: изменить число детей у вершин дерева

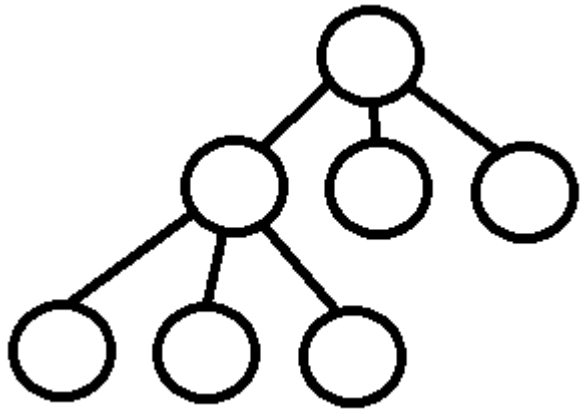
- Бинарная куча – частный случай, k=2

Что нам даст болшее k?

- Высота дерева будет меньше

  $T_{SiftUp} = O(\log_k N)$

  $T_{SiftDown} = O(k \log_k N)$

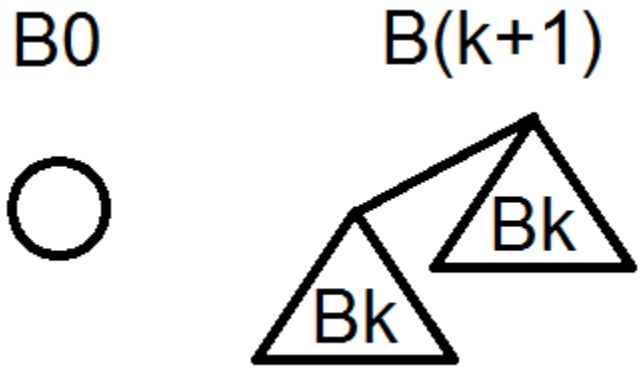


«Сливаемые кучи»

Цель: добавить операцию merge: (куча, куча) -> куча

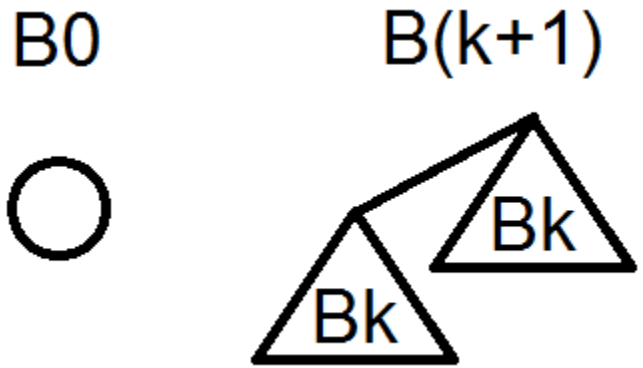
Биномиальная куча

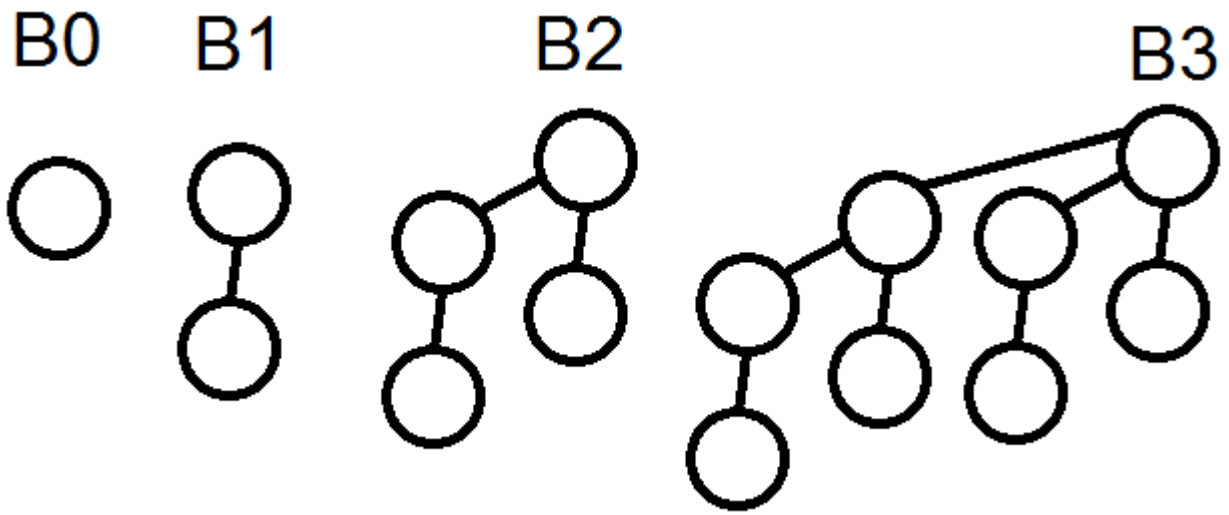
Определим биномиальное дерево



Биномиальная куча

Определим биномиальное дерево

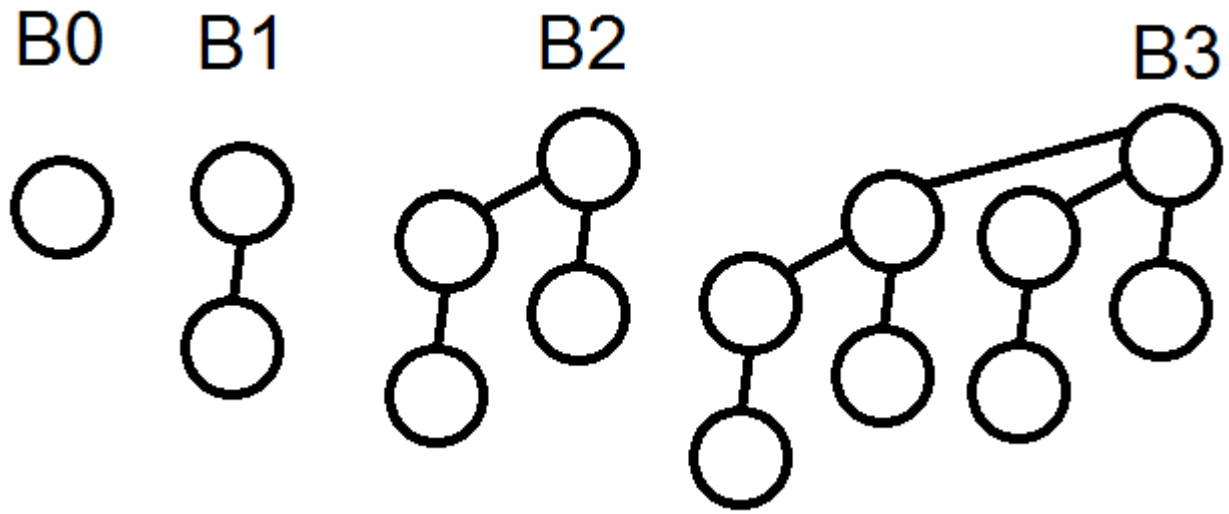




Биномиальная куча

Свойства биномиальных деревьев:

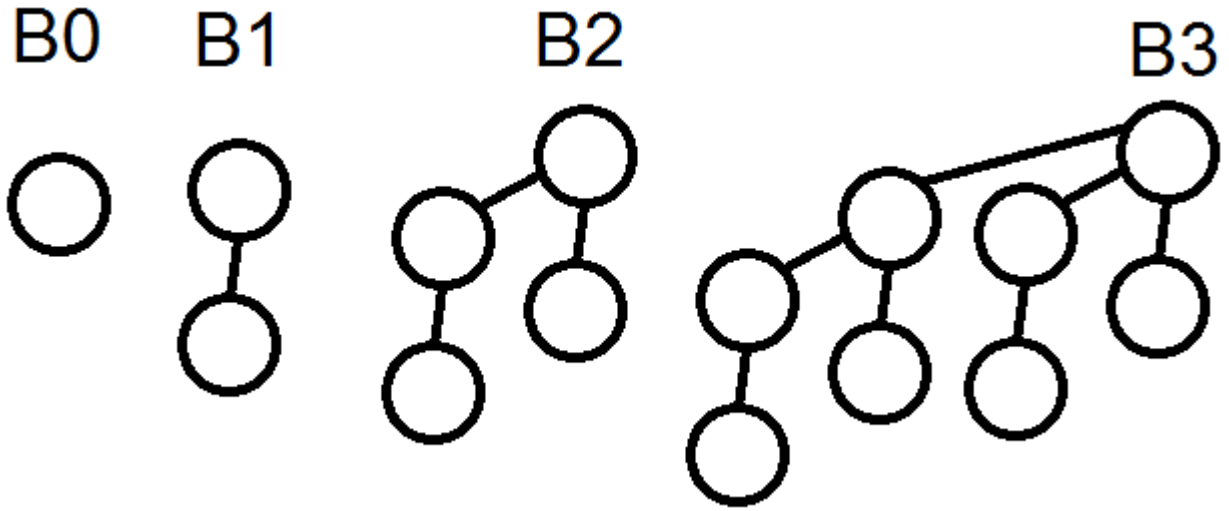
- B_K имеет 2^K вершин



Биномиальная куча

Свойства биномиальных деревьев:

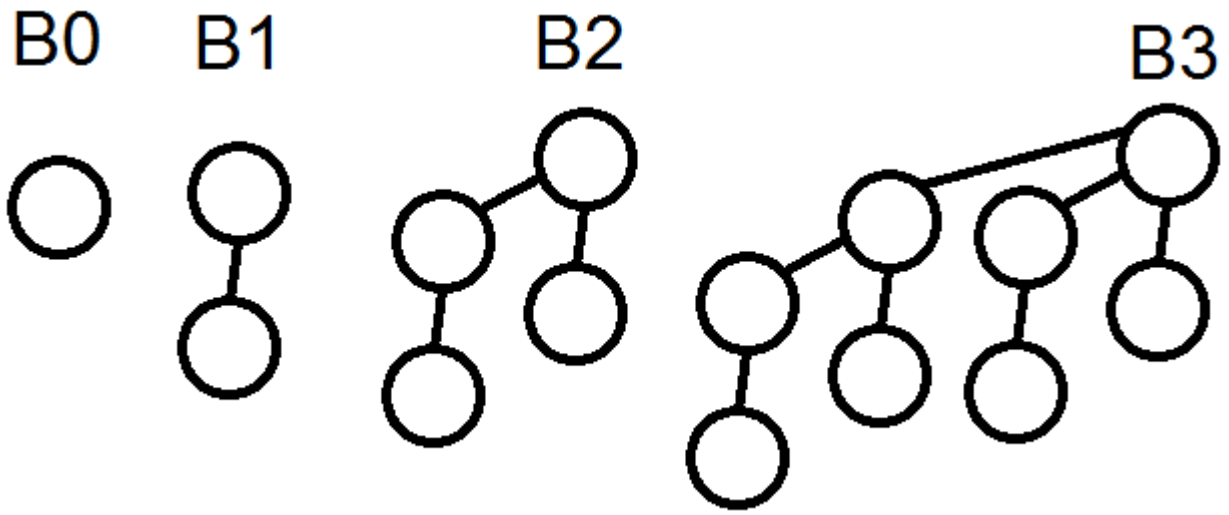
- B_K имеет 2^K вершин
- Высота B_K = K



Биномиальная куча

Свойства биномиальных деревьев:

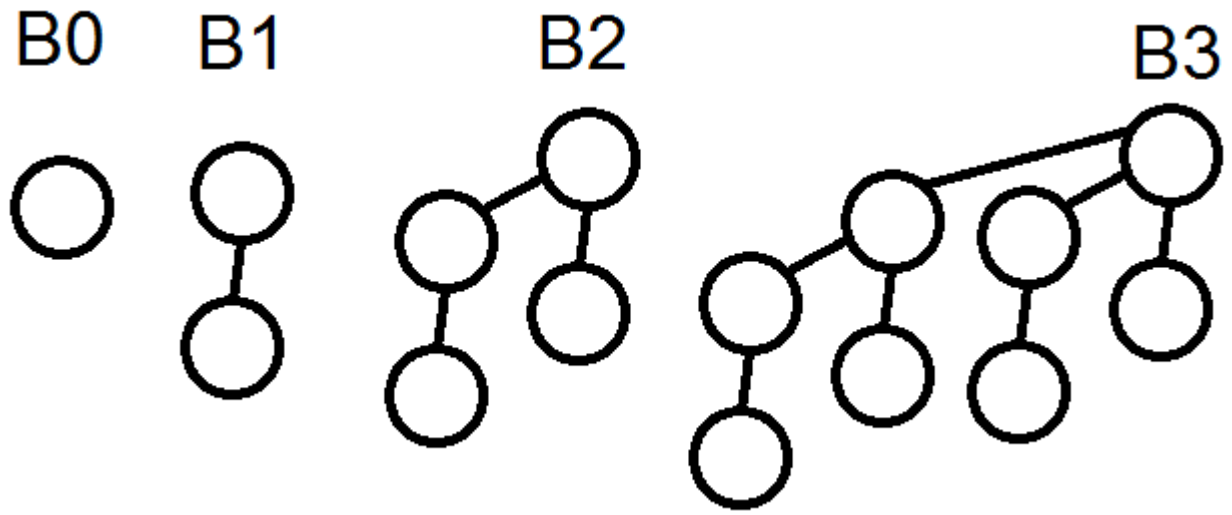
- B_K имеет 2^K вершин
- Высота B_K = K
- C(i,k) вершин на i-ом слое



Биномиальная куча

Свойства биномиальных деревьев:

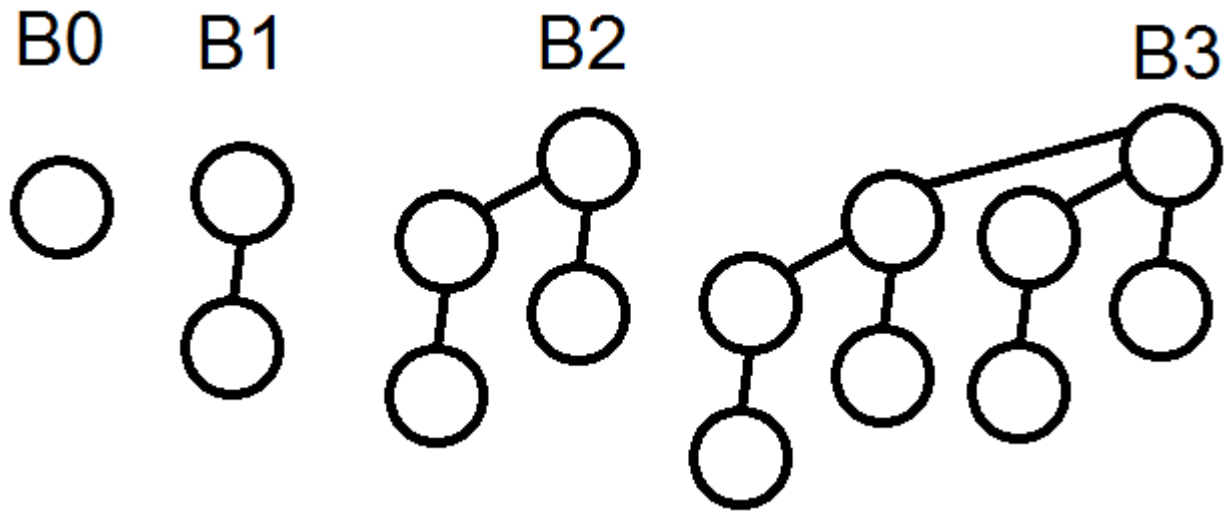
- B_K имеет 2^K вершин
- Высота B_K = K
- C(i,k) вершин на i-ом слое
- У корня B_K K детей



Биномиальная куча

Свойства биномиальных деревьев:

- B_K имеет 2^K вершин
- Высота B_K = K
- C(i,k) вершин на i-ом слое
- У корня B_K K детей
- У корня максимальное число детей



Биномиальная куча

Биномиальная куча – набор биномиальных деревьев

Биномиальная куча

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи

Биномиальная куча

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

H1 = {B0, B1, B2}

H2 = {B1, B4}

merge(H1, H2) -> {B0, B1, B1, B2, B4}

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

H1 = {B0, B1, B2}

H2 = {B1, B4}

merge(H1, H2) -> {B0, B1, B1, B2, B4}

Два дерева высоты 1

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

H1 = {B0, B1, B2}

H2 = {B1, B4}

merge(H1, H2) -> {B0, B1, B1, B2, B4}

Два дерева высоты 1, сливаем:

- подвешиваем корень одного к корню другого

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

H1 = {B0, B1, B2}

H2 = {B1, B4}

merge(H1, H2) -> {B0, B1, B1, B2, B4}

merge(H1, H2) -> {B0, B2, B2, B4}

Два дерева высоты 1, сливаем:

- подвешиваем корень одного к корню другого
- получем из них дерево высоты 2

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

H1 = {B0, B1, B2}

H2 = {B1, B4}

merge(H1, H2) -> {B0, B2, B2, B4}

merge(H1, H2) -> {B0, B3, B4}

Биномиальная куча: merge

Биномиальная куча – набор биномиальных деревьев

- Каждое дерево обладает свойсвом кучи
- Все деревья разного размера

Как слить две кучи?

H1 = {B0, B1, B2}

H2 = {B1, B4}

merge(H1, H2) -> {B0, B3, B4}

Биномиальная куча: merge

Слияние биномиальных куч похоже на суммирование бинарных чисел в столбик с переносом в старший разряд

Биномиальная куча: merge

Слияние биномиальных куч похоже на суммирование бинарных чисел в столбик с переносом в старший разряд

- Время на слияние = число битов

Биномиальная куча: merge

Слияние биномиальных куч похоже на суммирование бинарных чисел в столбик с переносом в старший разряд

- Время на слияние = число битов = O(logN)

Биномиальная куча: операции

Другие операции через слияние

- add

- extractMin

Биномиальная куча: операции

Другие операции через слияние

- add
  - слияние с кучей из одного элемента

- extractMin

Биномиальная куча: операции

Другие операции через слияние

- add
  - слияние с кучей из одного элемента
  - T = O(logN)
- extractMin

Биномиальная куча: операции

Другие операции через слияние

- add
  - слияние с кучей из одного элемента
  - T = O(logN)
- extractMin
  - одно дерево распадается

Биномиальная куча: операции

Другие операции через слияние

- add
  - слияние с кучей из одного элемента
  - T = O(logN)
- extractMin
  - одно дерево распадается
  - вместо этого дерева получается другая биномиальная куча

Биномиальная куча: операции

Другие операции через слияние

- add
  - слияние с кучей из одного элемента
  - T = O(logN)
- extractMin
  - одно дерево распадается
  - вместо этого дерева получается другая биномиальная куча
  - сливаем две кучи

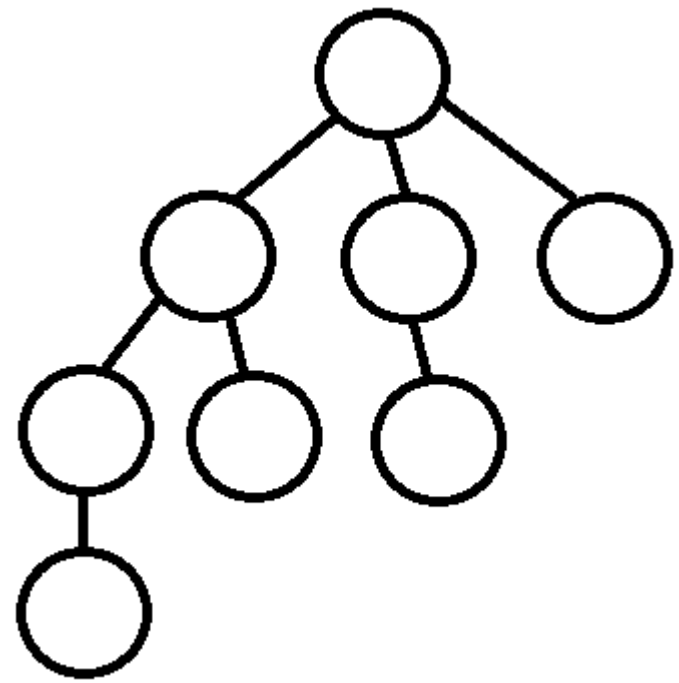
Биномиальная куча: операции

Другие операции через слияние

- add
  - слияние с кучей из одного элемента
  - T = O(logN)
- extractMin
  - одно дерево распадается
  - вместо этого дерева получается другая биномиальная куча
  - сливаем две кучи
  - T = O(logN)

Биномиальная куча: советы по реализации

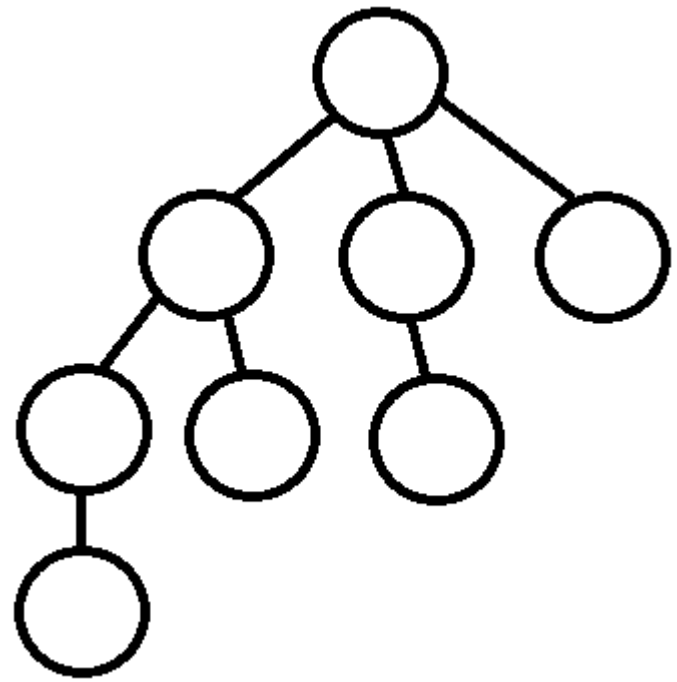
Как организовать в памяти структуру биномиального дерева?



Биномиальная куча: советы по реализации

Как организовать в памяти структуру биномиального дерева?

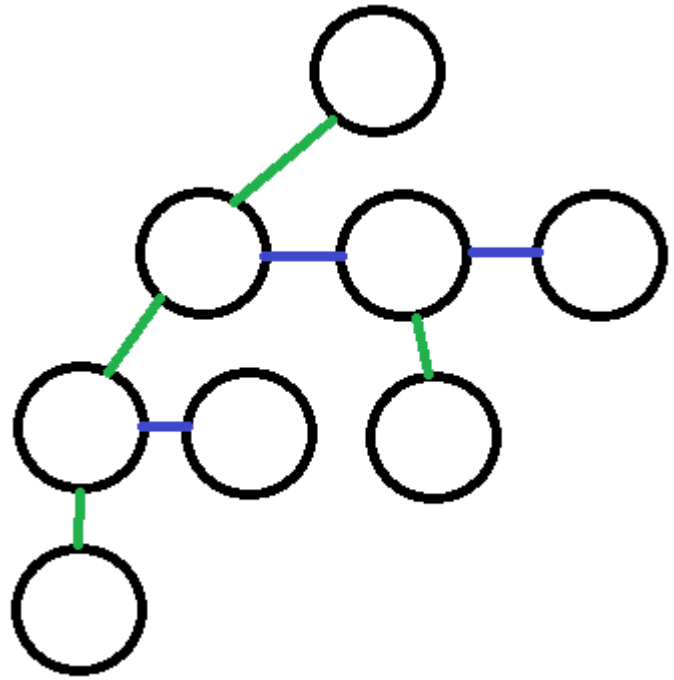
- Из вершины исходит неограниченное число рёбер



Биномиальная куча: советы по реализации

Как организовать в памяти структуру биномиального дерева?

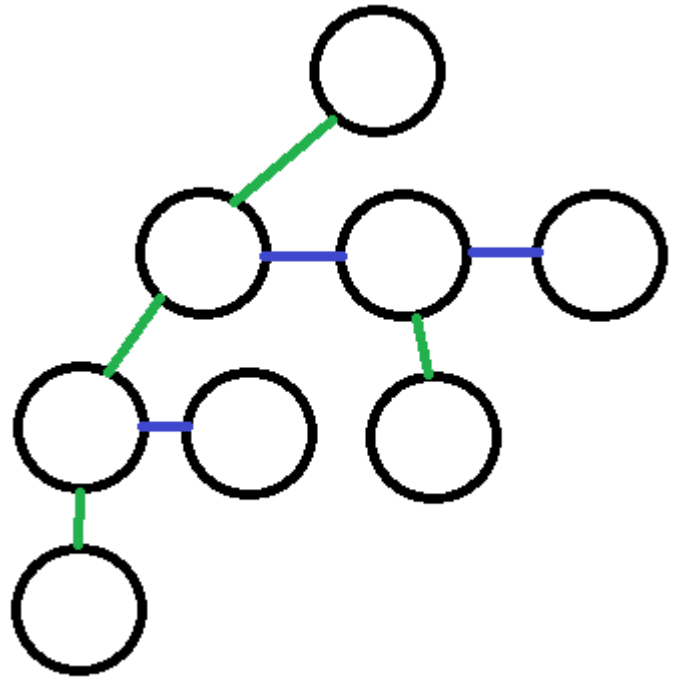
- ~~Из вершины исходит неограниченное число рёбер~~
- В каждой вершине храним две ссылки



Биномиальная куча: советы по реализации

Как организовать в памяти структуру биномиального дерева?

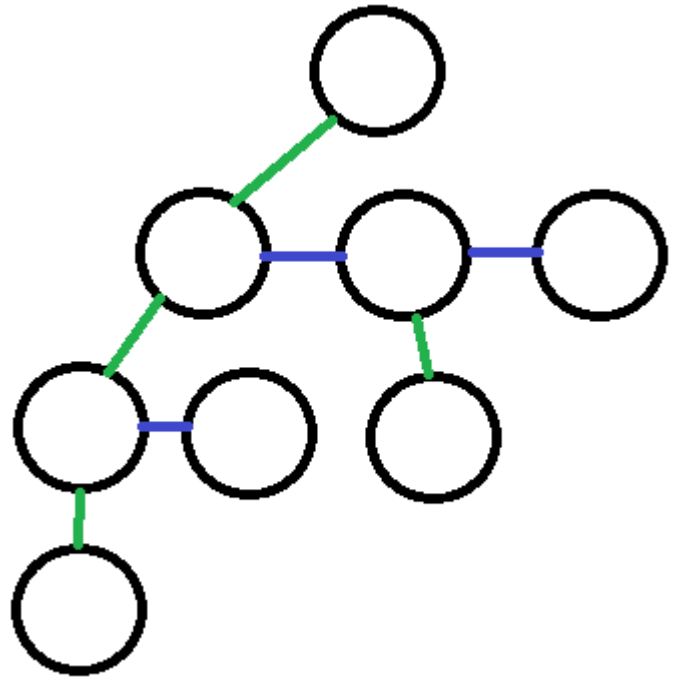
- ~~Из вершины исходит неограниченное число рёбер~~
- В каждой вершине храним две ссылки:
  - Первый ребёнок



Биномиальная куча: советы по реализации

Как организовать в памяти структуру биномиального дерева?

- ~~Из вершины исходит неограниченное число рёбер~~
- В каждой вершине храним две ссылки:
  - Первый ребёнок
  - «Брат справа»



Левацкая (левосторонняя) куча | Leftist heap

Иной способ организовать сливаемые кучи

Левацкая (левосторонняя) куча | Leftist heap

Иной способ организовать сливаемые кучи

Куча на одном бинарном, но не сбалансированном дереве

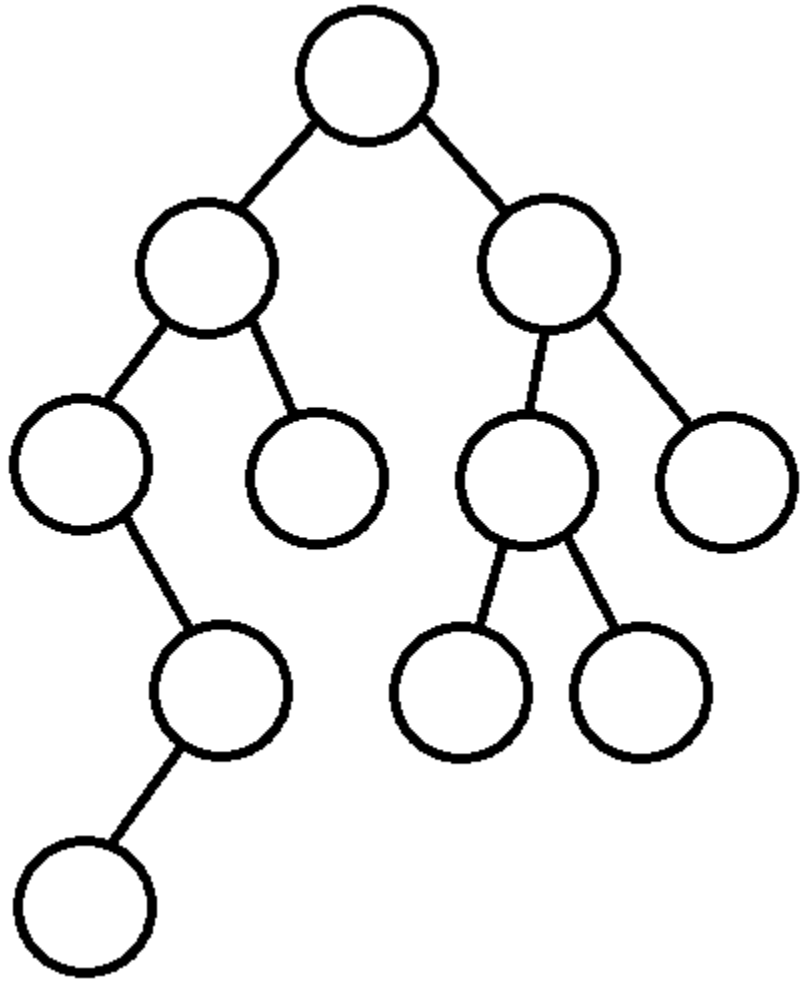
Левацкая куча: определение

Определим ранг вершины

Левацкая куча: определение

Определим ранг вершины

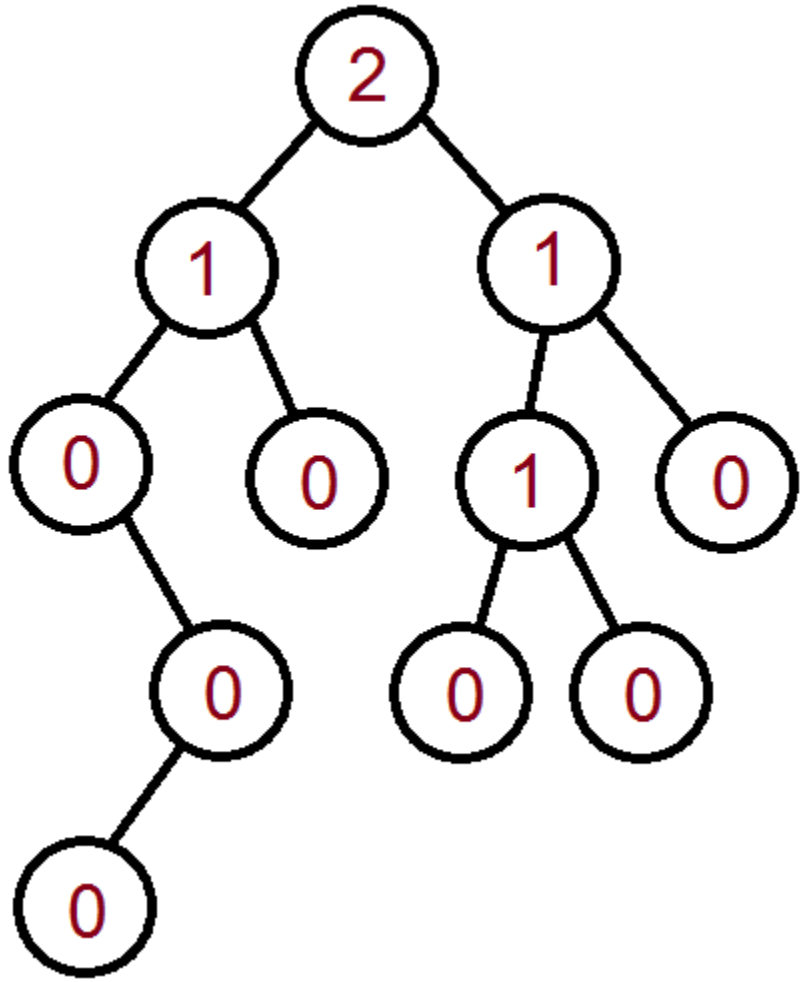
- Ранг вершины = глубина до ближайшего остутствующего потомка



Левацкая куча: определение

Определим ранг вершины

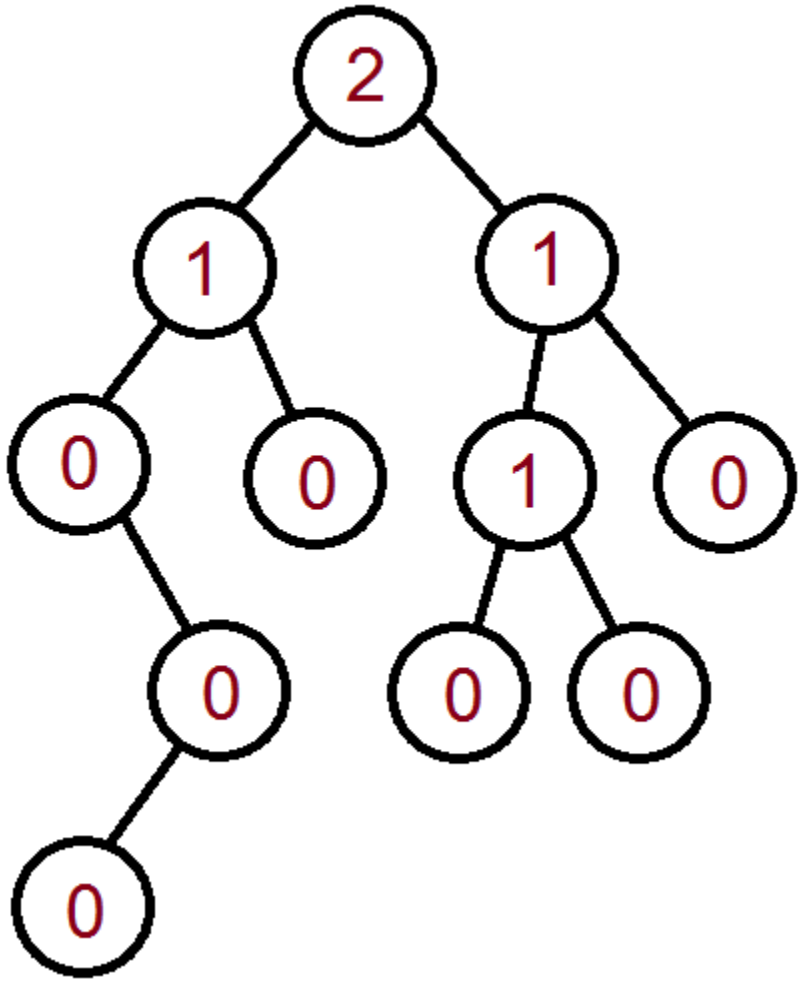
- Ранг вершины = глубина до ближайшего остутствующего потомка



Левацкая куча: определение

Определим ранг вершины

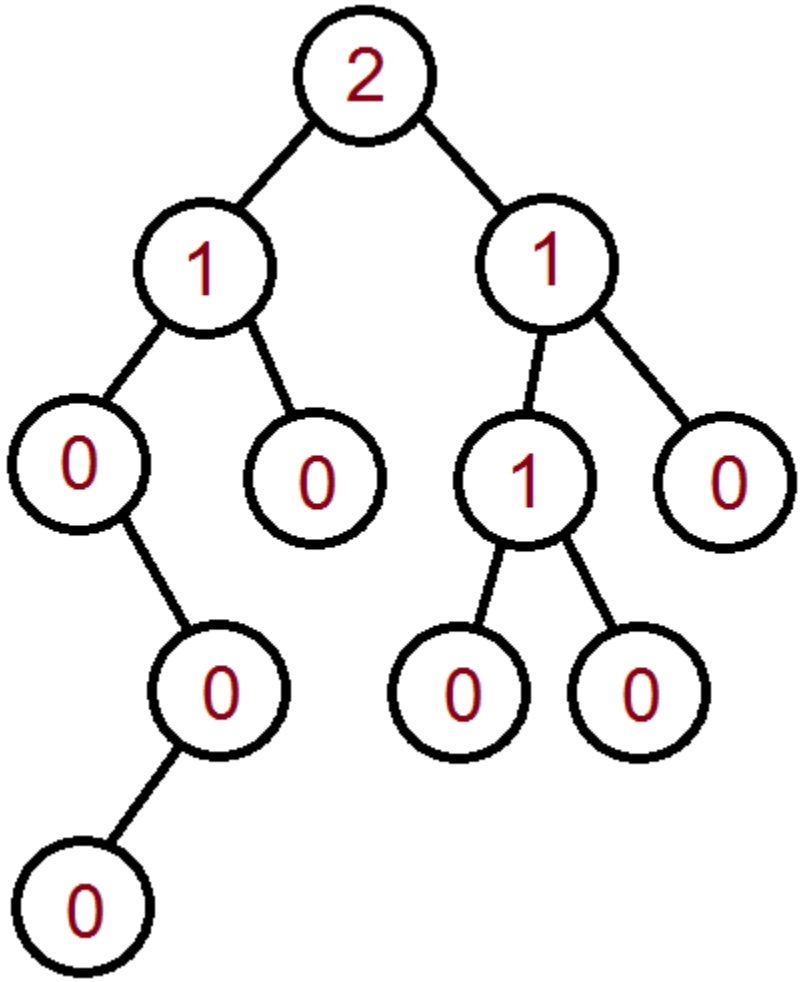
- Ранг вершины = глубина до ближайшего остутствующего потомка
  - Rk(v) = min(Rk(l), Rk(r)) + 1



Левацкая куча: определение

Определим ранг вершины

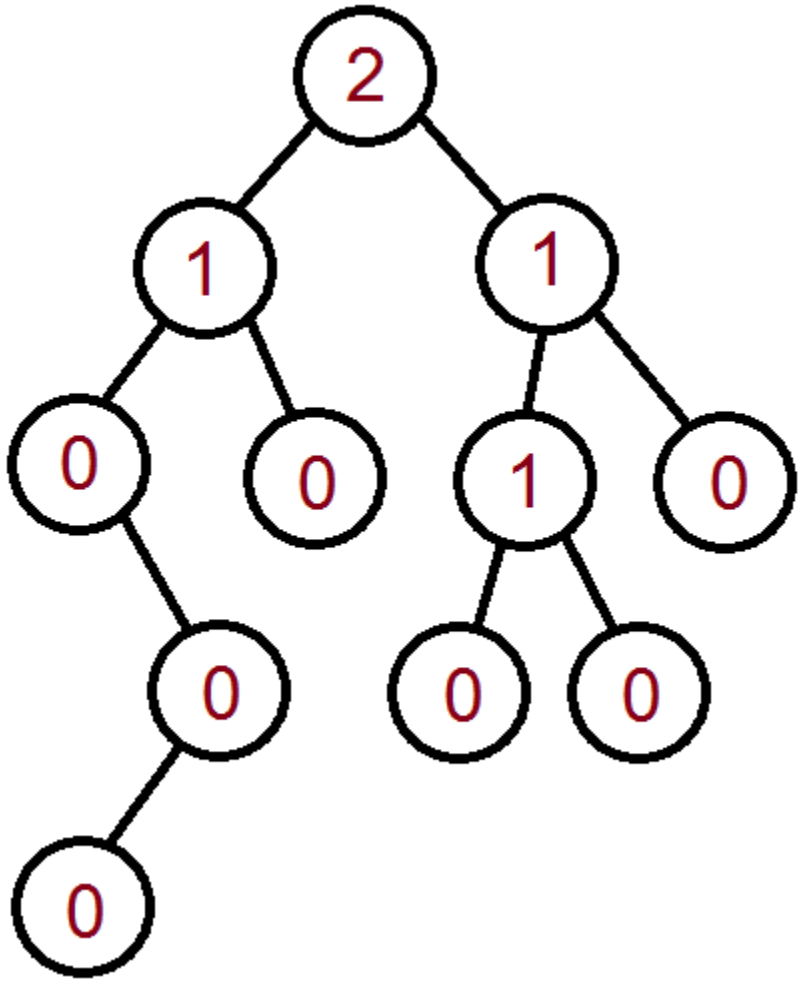
- Ранг вершины = глубина до ближайшего остутствующего потомка
  - Rk(v) = min(Rk(l), Rk(r)) + 1
  - Rk(null) = -1



Левацкая куча: определение

Определим ранг вершины

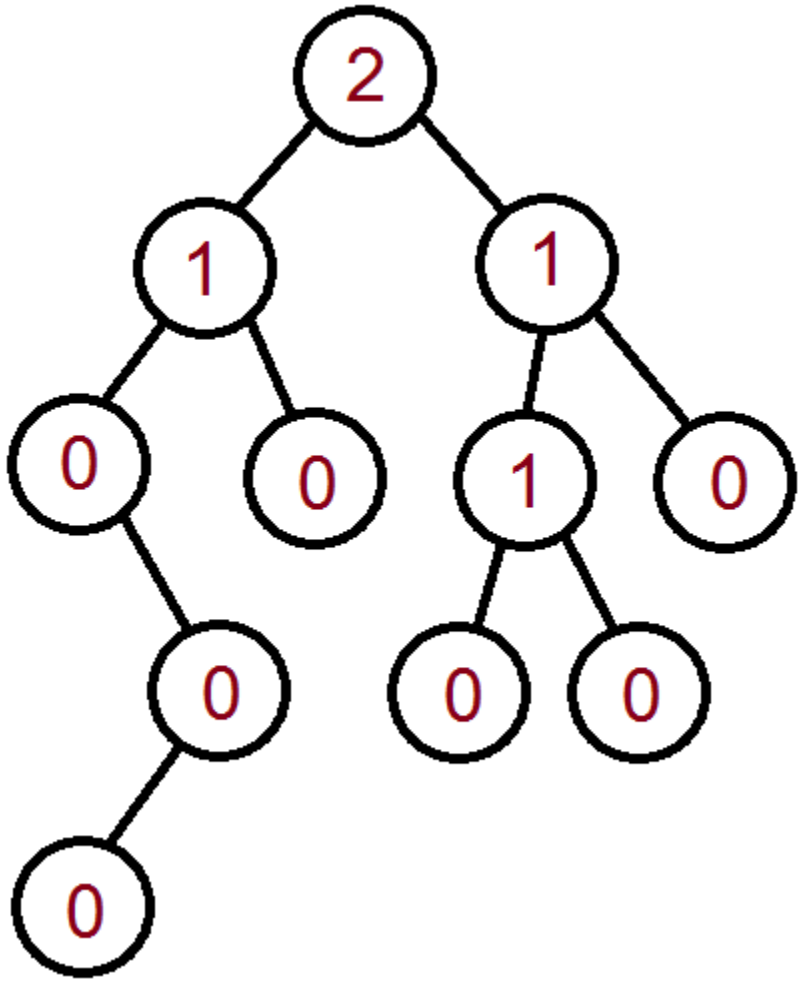
- Ранг вершины = глубина до ближайшего остутствующего потомка
  - Rk(v) = min(Rk(l), Rk(r)) + 1
  - Rk(null) = -1
- Свойство ранга:
  - размер поддерева v ≥ 2^Rk(v)
  - иначе говоря, Rk(v) ≤ log(размера)



Левацкая куча: определение

Определение:

- дерево левацкое, если везде Rk(l) ≥ Rk(r)

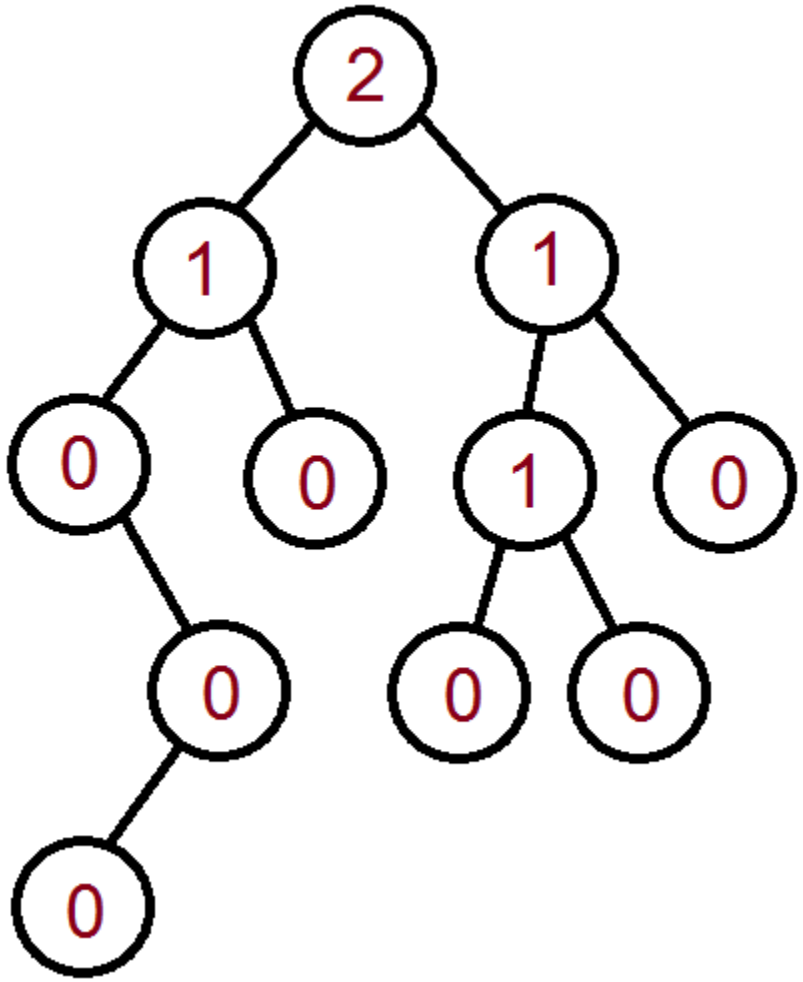


Левацкая куча: определение

Определение:

- дерево левацкое, если везде Rk(l) ≥ Rk(r)

Это дерево левацкое?

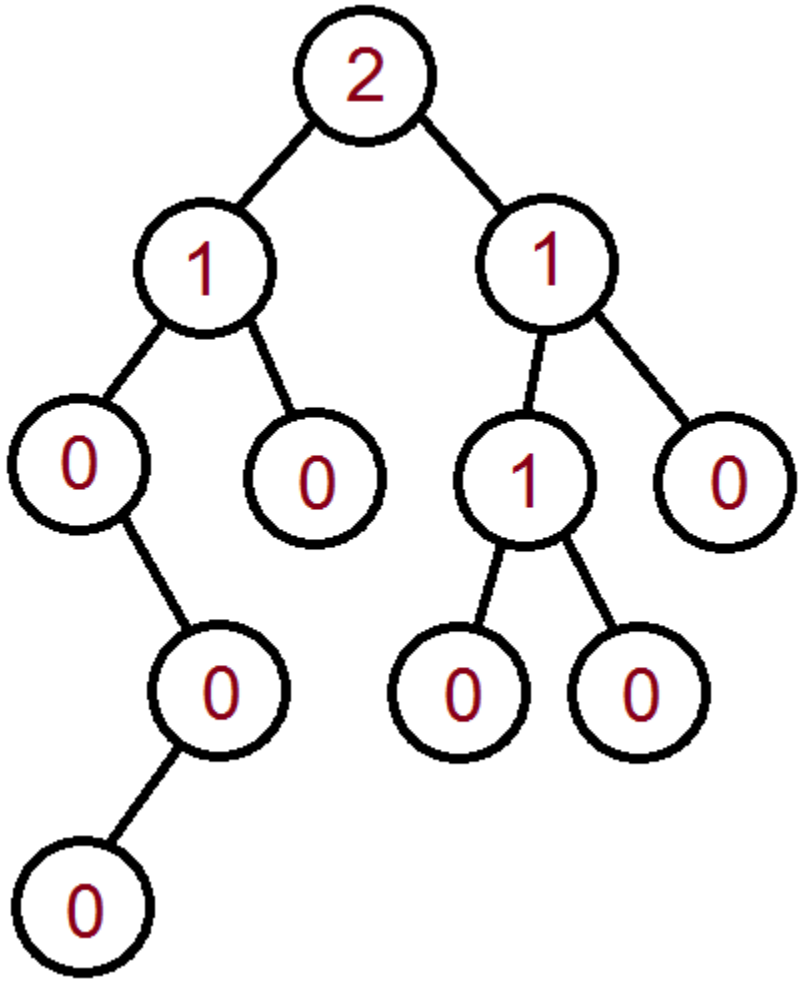


Левацкая куча: определение

Определение:

- дерево левацкое, если везде Rk(l) ≥ Rk(r)

Дерево можно сделать левацким



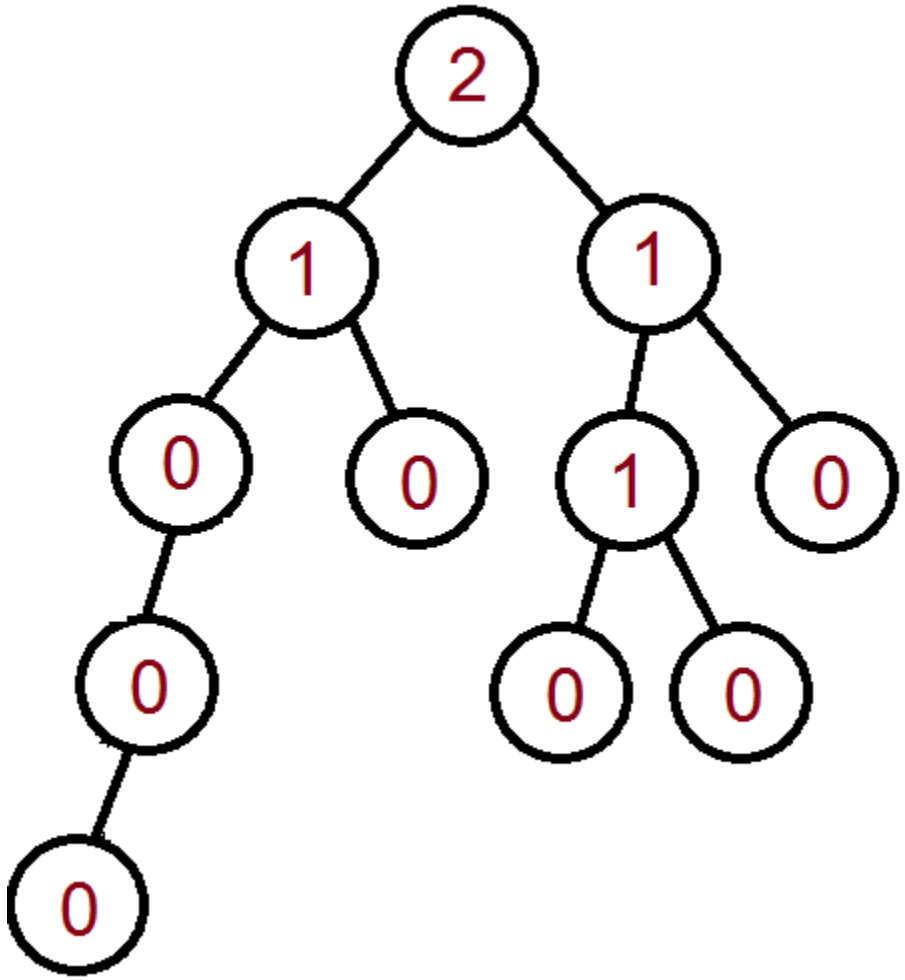
Левацкая куча: определение

Определение:

- дерево левацкое, если везде Rk(l) ≥ Rk(r)

Дерево можно сделать левацким,

поменяв l и r, там где условие нарушено



Левацкая куча: определение

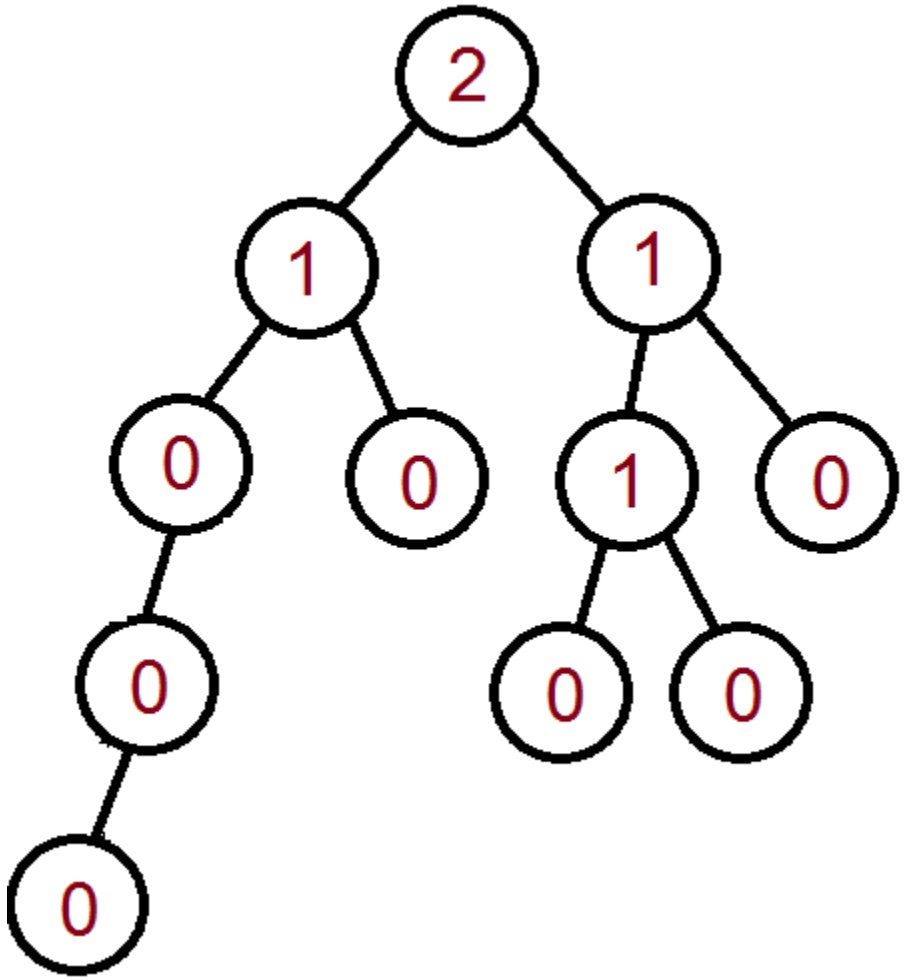
Определение:

- дерево левацкое, если везде Rk(l) ≥ Rk(r)

Дерево можно сделать левацким,

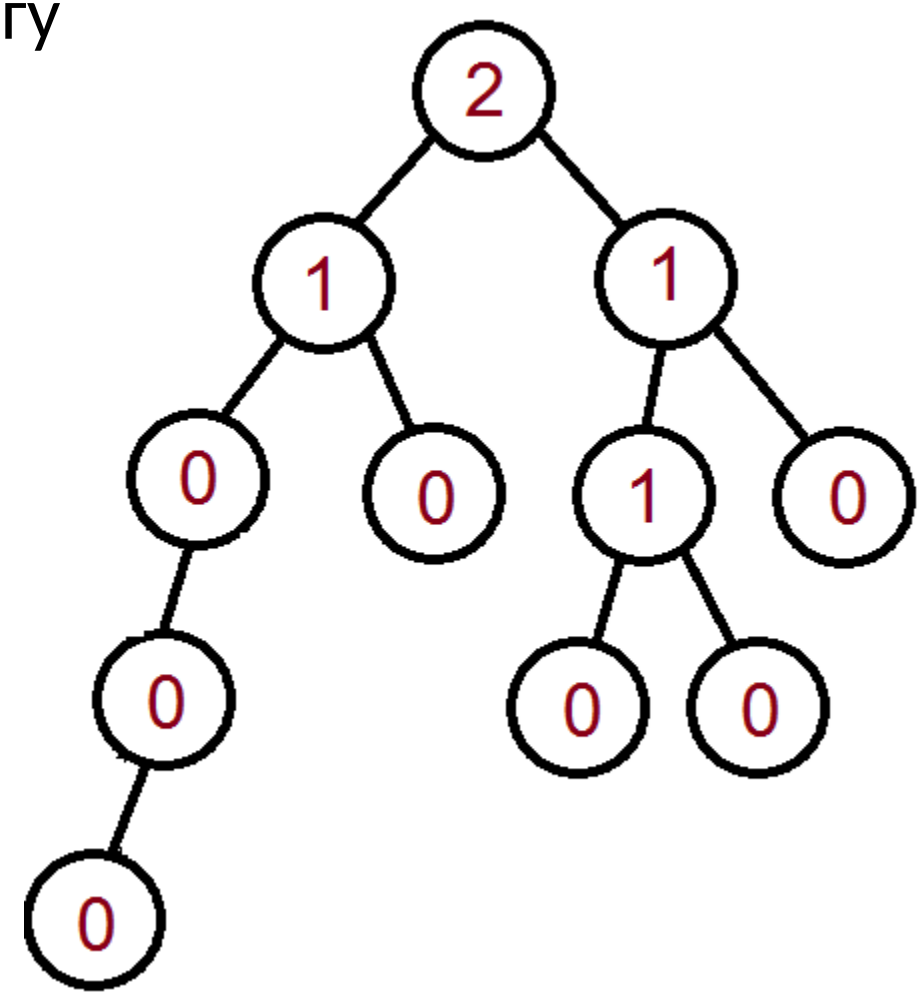
поменяв l и r, там где условие нарушено

Левацкая куча – куча на левацком дереве



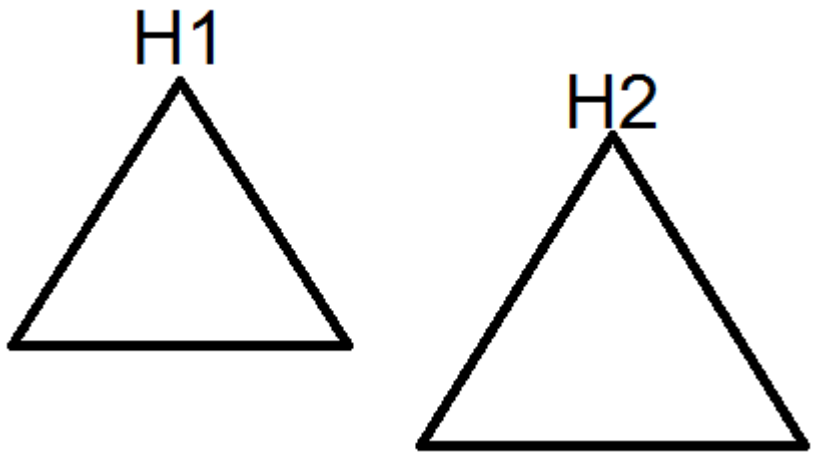
Левацкая куча: свойство

Длина правого пути в левацком дереве = рангу



Левацкая куча: merge

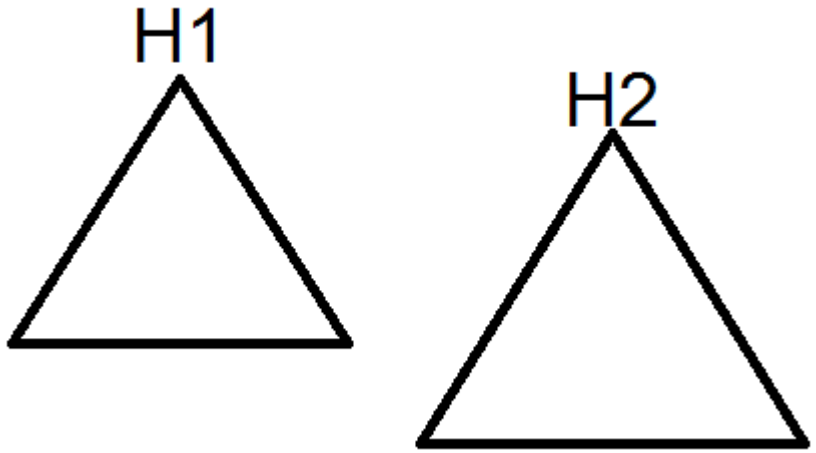
Сливаем пару левацких куч, H1 и H2



Левацкая куча: merge

Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

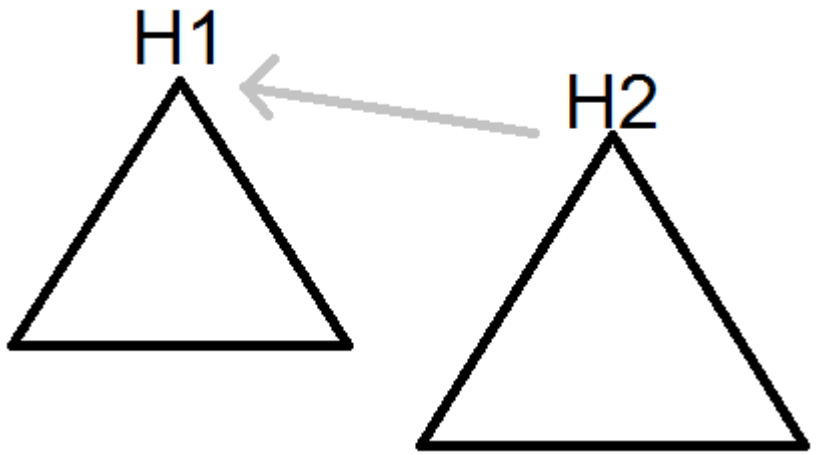


Левацкая куча: merge

Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

- Значит H2 подвесим к H1



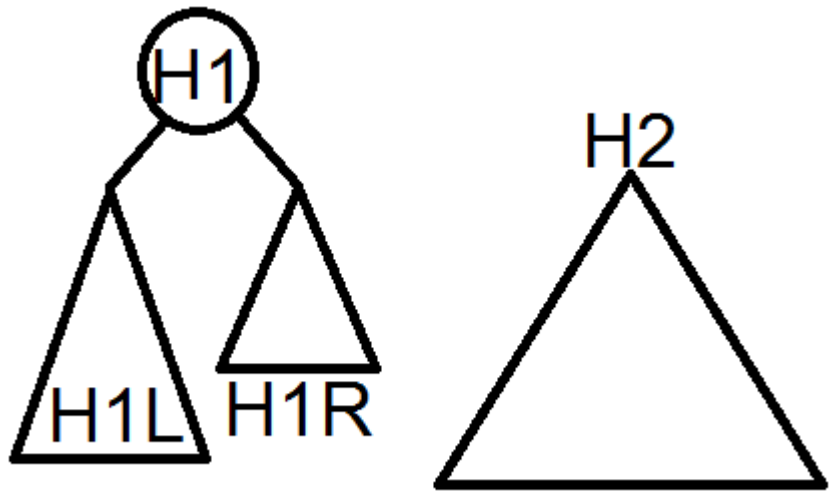
Левацкая куча: merge

Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

- Значит H2 подвесим к H1

Некуда подвешивать?



Левацкая куча: merge

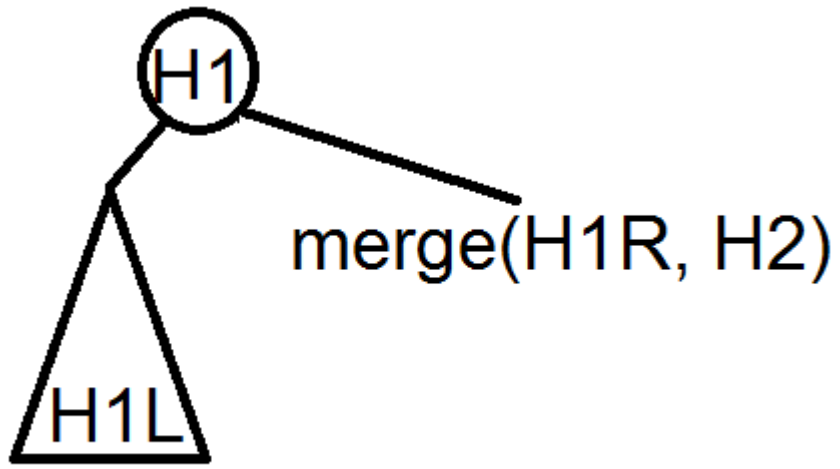
Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

- Значит H2 подвесим к H1

Некуда подвешивать?

- Сольём правого сына H1 с H2



Левацкая куча: merge

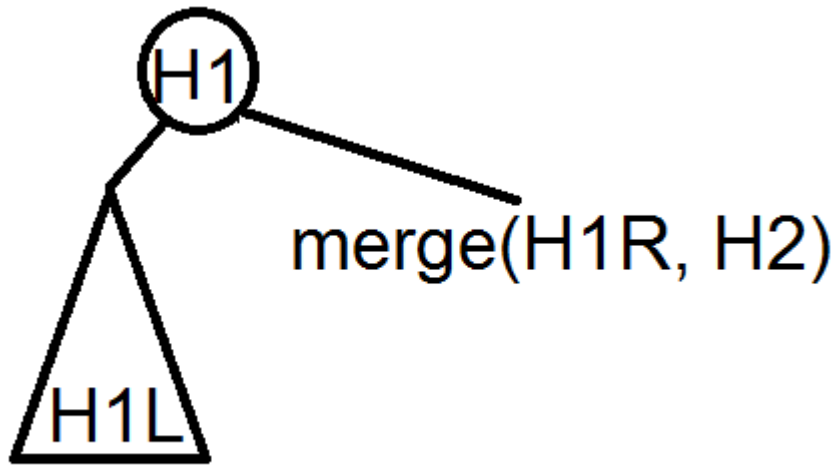
Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

- Значит H2 подвесим к H1

Некуда подвешивать?

- Сольём правого сына H1 с H2
- Результат подвесим справа к H1



Левацкая куча: merge

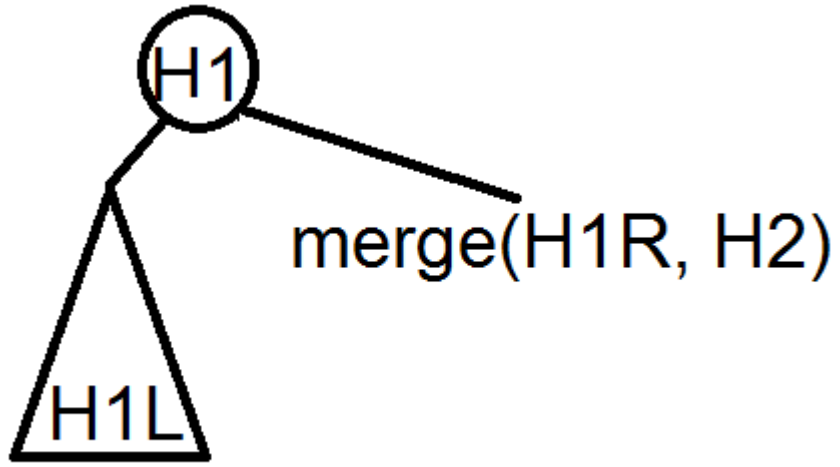
Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

- Значит H2 подвесим к H1

Некуда подвешивать?

- Сольём правого сына H1 с H2
- Результат подвесим справа к H1



В корне H1 могло нарушиться свойство левацкости

Левацкая куча: merge

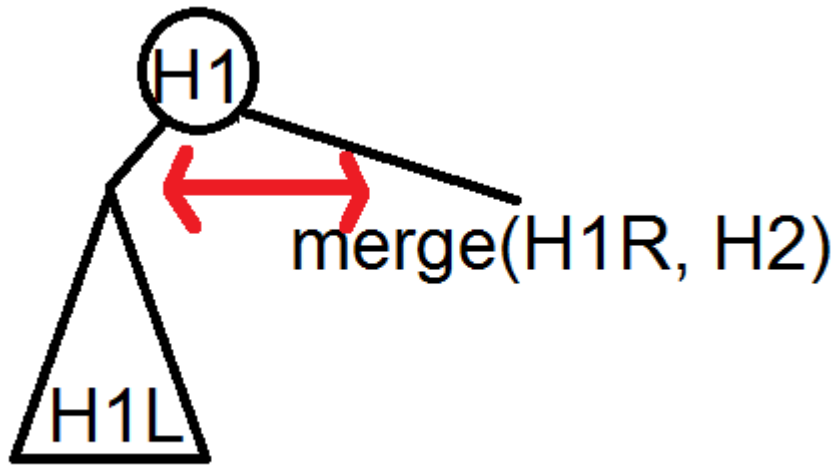
Сливаем пару левацких куч, H1 и H2

Пусть min(H1) ≤ min(H2)

- Значит H2 подвесим к H1

Некуда подвешивать?

- Сольём правого сына H1 с H2
- Результат подвесим справа к H1



В корне H1 могло нарушиться свойство левацкости

- Тогда делаем swap его детей

Левацкая куча: сложность merge

- Merge проходит по правым ветвям двух деревьев

Левацкая куча: сложность merge

- Merge проходит по правым ветвям двух деревьев
- T = O(logN)

Левацкая куча: операции

- Add

Левацкая куча: операции

- Add:
  - Сливаем с кучей из одной вершины

Левацкая куча: операции

- Add:
  - Сливаем с кучей из одной вершины
- ExtractMin

Левацкая куча: операции

- Add:
  - Сливаем с кучей из одной вершины
- ExtractMin:
  - Отрезаем корень, куча распадётся на две части

Левацкая куча: операции

- Add:
  - Сливаем с кучей из одной вершины
- ExtractMin:
  - Отрезаем корень, куча распадётся на две части
  - Сливаем их

Левацкая куча: сложность

- $T_{Add} = O(logN)$
- $T_{ExtractMin} = O(logN)$

Левацкая куча: сложность

- $T_{Add} = O(logN)$
- $T_{ExtractMin} = O(logN)$
- $T_{SiftUp} = O(N)$

Левацкая куча: сложность

- $T_{Add} = O(logN)$
- $T_{ExtractMin} = O(logN)$
- $T_{SiftUp} = O(N)$ – не стоит делать SiftUp

Косая куча | Skew heap

- Что, если отказаться запоминать ранг для вершин левацкой кучи?

Косая куча | Skew heap

- Что, если отказаться запоминать ранг для вершин левацкой кучи?
- Зачем ранг?

Косая куча | Skew heap

- Что, если отказаться запоминать ранг для вершин левацкой кучи?
  - Зачем ранг?
    - Ранг определяет, когда делать swap

Косая куча | Skew heap

- Что, если отказаться запоминать ранг для вершин левацкой кучи?
  - Зачем ранг?
    - Ранг определяет, когда делать swap
- Давайте просто всегда делать swap

Косая куча: сложность merge

Нужно ввести потенциал на косой куче

Косая куча: сложность merge

Нужно ввести потенциал на косой куче

- Опредим вес вершины как число элементов в её поддереве

Косая куча: сложность merge

Нужно ввести потенциал на косой куче

- Опредим вес вершины как число элементов в её поддереве
- Тяжёлая вершина имеет вес > половина веса её родителя

Косая куча: сложность merge

Нужно ввести потенциал на косой куче

- Опредим вес вершины как число элементов в её поддереве
- Тяжёлая вершина имеет вес > половина веса её родителя
- Если тяжёлая вершина является правым сыном, то она плохая

Косая куча: сложность merge

Нужно ввести потенциал на косой куче

- Опредим вес вершины как число элементов в её поддереве
- Тяжёлая вершина имеет вес > половина веса её родителя
- Если тяжёлая вершина является правым сыном, то она плохая
- Потенциал = число плохих вершин

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Merge проходит по правому пути

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Merge проходит по правому пути
- На правом пути число не плохих вершин = O(logN)

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Merge проходит по правому пути
- На правом пути число не плохих вершин = O(logN)
  - Если вершина не плохая, то число вершин в поддереве уменьшается в два раза

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Merge проходит по правому пути
- На правом пути число не плохих вершин = O(logN)
  - Если вершина не плохая, то число вершин в поддереве уменьшается в два раза
- Плохие вершины с правого пути перемещаются на левый путь

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Merge проходит по правому пути
- На правом пути число не плохих вершин = O(logN)
  - Если вершина не плохая, то число вершин в поддереве уменьшается в два раза
- Плохие вершины с правого пути перемещаются на левый путь
  - Уменьшение потенциала оплачивает перемещение

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Насколько потенциал увеличится?

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Насколько потенциал увеличится?
- Каждая возникшая после swap плохая вершина уменьшает число вершин в левом поддереве в два раза

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Насколько потенциал увеличится?
- Каждая возникшая после swap плохая вершина уменьшает число вершин в левом поддереве в два раза
  - Возникнет O(logN) плохих вершин

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Насколько потенциал увеличится?
- Каждая возникшая после swap плохая вершина уменьшает число вершин в левом поддереве в два раза
  - Возникнет O(logN) плохих вершин
- Т.е. потенциал возрастёт на O(N)

Косая куча: сложность merge

Если тяжёлая вершина является правым сыном, то она плохая

Потенциал = число плохих вершин

- Насколько потенциал увеличится?
- Каждая возникшая после swap плохая вершина уменьшает число вершин в левом поддереве в два раза
  - Возникнет O(logN) плохих вершин
- Т.е. потенциал возрастёт на O(N)
  - Merge за O(logN) в учётном смысле