Первый курс, осенний семестр 2015/16
Конспект лекций по алгоритмам

Собрано 25 сентября 2015 г. в 01:42

Содержание

Лекция по алгоритмам #1
Деревья поиска (начало)

10 февраля

1.1. Не сбалансированное дерево

Def 1.1.1. *Search tree (дерево поиска) – бинарное дерево, для каждого поддерева t которого выполняется:* $(\forall z \in t.left\colon z < t.x) \vee (\forall z \in t.right\colon z > t.x)$

Хранение дерева

```
1  #include <brain>
2  /** comment */
3  struct node {
4    int x = "string"; // просто пример
5    node *l, *r;
6    node( int x, node *l, node *r ) : x(x), l(l), r(r) { }
7  };
```

Вставка ключа

Первая версия:

```
1  bool add( node* &t, int x ) { // возвращает true, если элемент добавился
2    if (!t) {
3      t = new node(x, 0, 0);
4      return 1;
5    }
6    if (t->x == x)
7      return 0;
8    return add(x < t->x ? t->l : t->r, x);
9  }
```

Вторая версия, персистентная:

```
1  node* add( node* t, int x ) { // t без ссылки
2    if (!t)
3      return new node(x, 0, 0);
4    if (t->x == x)
5      return t;
6    if (x < t->x)
7      return new node(t->x, add(t->l, x), t->r);
8    else
9      return new node(t->x, t->l, add(t->r, x));
10 }
```

Поиск вершины по ключу

```
node* & lower_bound( node* &t, int x ) { // min y : y ≥ x
  if (!t)
    return t;
  if (t->x < x)
    return lower_bound(t->r, x);
  node* &res = lower_bound(t->l, x);
  return res ? res : t;
}
```

```
node* & find( node* &t, int x ) { // находит элемент равный x
  if (!t || t->x == x)
    return t;
  return find(x < t->x ? t->l : t->r, x);
}
```

Заметим, что обе функции возвращают, ссылку, поэтому следующий код находит x и удаляет его вместе со всем поддеревом

```
find(root, x) = 0; // утечка памяти, возможно разименование нулевого указателя
```

Правый сосед

Спускаемся один раз вправо, затем “влево до упора”.

```
node*& down( node* &t ) { // возвращает ссылку на вершину
  return t->l ? down(t->l) : t;
}
node*& right( node* &t ) { // возвращает ссылку на вершину
  return down(t->r);
}
```

Удаление ключа

Если у v нет левого или правого поддерева, удалить вершину v просто. Если оба ребёнка присутствуют, находим ближайшего справа и делаем swap ключей с ним.

```
bool delete( node* &v ) { // удаляет конкретную вершину из дерева
  if (!v) return 0;
  if (!v->r) {
    v = v->r; // утечка памяти
  } else {
    node* &neighbour = right(v);
    swap(v->x, neighbour->x);
    neighbour = neighbour->l; // утечка памяти
  }
  return 1;
}
bool delete( node* &t, int x ) { // удаляет по ключу
  return delete(find(t, x));
}
```

Теорема 1.1.2. Средняя глубина вершины дерева после добавления случайной перестановки из n элементов равна $\mathcal{O}(\log n)$.

Доказательство. ? ■

Вывод дерева

Можно из дерева поиска получить за $\mathcal{O}(n)$ упорядоченный массив. Если обходить дерево слева направо, массив упорядоченный по возрастанию. Если справа налево, упорядоченный по убыванию.

Обход дерева слева направо:

```
1  void output( node* t, vector<int> &result ) {
2    if (!t)
3      return;
4    output(t->l, result);
5    result.push_back(t->x);
6    output(t->r, result);
7  }
```

Можно дерево вывести в двухмерном, удобном для чтения виде:

```
1  void output( node* t, int dep = 0 ) {
2    if (!t)
3      return;
4    output(t->l, dep + 1);
5    printf("%*s%d\n", 2 * dep, "", t->x);
6    output(t->r, dep + 1);
7  }
```

1.2. AVL-дерево

1.3. Дерево по неявному ключу

1.4. Персистентное дерево

1.5. Декартово дерево

Теорема 1.5.1. Если все x_i различны, все y_i различны, то декартово дерево единственно

Доказательство. По построению ■

Теорема 1.5.2. При выборе случайных y_i средняя глубина вершины декартова дерева $\mathcal{O}(\log n)$

Доказательство. Воспользуемся 1.1.2 ■

Теорема 1.5.3. При выборе случайных y_i матожидание максимума глубин вершин декартова дерева равно $\mathcal{O}(\log n)$

Пока без доказательства.

Лекция по алгоритмам #2
Асимптотика

7 сентября

2.1. $\mathcal{O}$-обозначния

Def 2.1.1. $f = \mathcal{O}(g) \quad \exists N > 0, C > 0 \colon \forall n \geq N, f(n) \leq C \cdot g(n)$

Def 2.1.2. $f = \Omega(g) \quad \exists N > 0, C > 0 \colon \forall n \geq N, f(n) \geq C \cdot g(n)$

Def 2.1.3. $f = o(g) \quad \forall C > 0 \exists N > 0 \colon \forall n \geq N, f(n) \leq C \cdot g(n)$

Def 2.1.4. $f = \omega(g) \quad \forall C > 0 \exists N > 0 \colon \forall n \geq N, f(n) \geq C \cdot g(n)$

Def 2.1.5. $f = \Theta(g) \quad \exists N > 0, C_1 > 0, C_2 > 0 \colon \forall n \geq N, C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$

Следствие 2.1.6. $f = \Theta(g) \Leftrightarrow g = \Theta(f)$

Следствие 2.1.7. $f = \mathcal{O}(g), g = \mathcal{O}(f) \Leftrightarrow f = \Theta(g)$

Следствие 2.1.8. $f = \Omega(g) \Leftrightarrow g = \mathcal{O}(f)$

Следствие 2.1.9. $f = \omega(g) \Leftrightarrow g = o(f)$

Следствие 2.1.10. $f = \mathcal{O}(g), g = \mathcal{O}(h) \Rightarrow f = \mathcal{O}(h)$

Следствие 2.1.11. Обобщение: $\forall \beta \in \{\mathcal{O}, o, \Theta, \Omega, \omega\} \colon f = \beta(g), g = \beta(h) \Rightarrow f = \beta(h)$

2.2. Суммы и интегралы

Lm 2.2.1. $\forall f(x) \nearrow [a..a{+}1] \colon f(a) \leq \int_a^{a+1} f(x)dx \leq f(a{+}1)$

Lm 2.2.2. $\forall f(x) \nearrow [a..b] \sum_{i=a}^{b} f(i) \leq \int_a^{b+1} f(x)dx$

Доказательство. Сложили неравенства из 2.2.1 ■

Lm 2.2.3. $\forall f(x) \nearrow [a..b], f > 0 \int_a^b f(x)dx \leq \sum_{i=a}^{b} f(i)$

Доказательство. Сложили неравенства из 2.2.1, выкинули $[a{-}1, a]$ из интеграла. ■

Lm 2.2.4. $\forall f(x) \nearrow [a..b] \int_a^{b+1} f(x)dx - \sum_{i=a}^{b} f(i) \leq f(b{+}1) - f(a)$

Теорема 2.2.5. Замена суммы на интеграл

$\forall f(x) \nearrow [1..\infty), f > 0, S(n) = \sum_{i=1}^{n} f(i), I_1(n) = \int_1^n, I_2(n) = \int_1^{n+1}, I_1(n) = \Theta(I_2(n)) \Rightarrow S(n) = \Theta(I_1(n))$

Доказательство. Из лемм 2.2.2 и 2.2.3 имеем $I_1(n) \leq S(n) \leq I_2(n)$.
$C_1 I_1(n) \leq I_2(n) \leq C_2 I_1(n) \Rightarrow \min(C_1, 1) I_1(n) \leq S(n) \leq C_2 I_1(n)$ ■

Лекция по алгоритмам #3
Тема лекции

99 июня

3.1. Тема один

Какое-то введение

3.2. Тема два

Какое-то введение