acm.mipt.ru

олимпиады по программированию на Физтехе

Декартово дерево: реализация линейного алгоритма построения на Си

- Теория о декартовом дереве
- Вход: Число элементов *n*, а затем *n* строчек пар (*x* , *y*). Пары должны идти в порядке возрастания ключей *x*.
- Выход: Префиксное изображение декартового дерева.

Пример использования:

```
bash$ gcc ct.c -o ct
bash$ ./ct
7
0 5
1 3
2 2
3 9
4 11
5 4
6 6
(4,11)
   (3,9)
      (0,5)
         ( nil )
         (1,3)
            ( nil )
            (2,2)
               ( nil )
               ( nil )
      ( nil )
   (6,6)
      (5,4)
         ( nil )
         ( nil )
      ( nil )
```

```c
/* file: ct.c
 * Cartesian Tree
 *
 */

#include <stdio.h>
#include <malloc.h>

typedef int dkey_t;

typedef struct node {
   struct node *l,*r,*p;
   dkey_t x;
   dkey_t y;
   int size;
} node_t;

int count_size(node_t *root) {
   if(root) {
      return (root->size = 1 + count_size(root->l) + count_size(root->r));
   } else {
      return 0;
   }
```

Поиск
Поиск
Раздел «Алгоритмы»
Главная
Форум
Ссылки
El Judge
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
}

node_t*
dt_new_node(dkey_t x, dkey_t y) {
   node_t *node = (node_t*)calloc(1, sizeof(node_t));
   node->x = x; node->y = y;
   return node;
}

node_t*
dt_add_right(node_t **root, node_t *old_right, dkey_t x, dkey_t y) {
   node_t *new_right = dt_new_node(x,y);
   while( old_right != *root && old_right->y < y ) {
      old_right = old_right->p;
   }
   if(old_right && old_right->y >= y) {
      new_right->l = old_right->r;
      old_right->r = new_right;
      new_right->p = old_right;
   } else {
      new_right->l = *root;
      *root = new_right;
   }
   return new_right;
}

void
prefix_traverse(node_t *root, void f(node_t*, int), int level) {
   if(root) {
      f(root, level);
      prefix_traverse(root->l, f, level+1);
      prefix_traverse(root->r, f, level+1);
   } else {
      f(root, level);
   }
}

void print_indented(node_t *node, int level) {
   int i;
   for(i=0; i < level; i++) printf("   ");
   if(node) printf("(%d,%d)\n", node->x, node->y);
   else     printf("( nil )\n");
}

int main(int argc, char *argv[])
{
   int n, i, x, y;
   node_t *root = 0;
   node_t *left = 0;
   scanf("%d", &n);
   for(i = 0 ; i < n ; i++) {
      scanf("%d%d", &x, &y);
      left = dt_add_left( &root, left, x, y );
   }
   prefix_traverse(root, print_indented, 0);
   count_size(root);
   while(scanf("%d%d", &x, &y) == 2) {
      printf("%d\n", count_lefter_and_lower(root, x, y, 0) );
   }

   return 0;
}
```

Подсчёт числа узлов в квадранте с углом (x,y)

С помощью декартовых деревьев можно, например, решать задачу 1028 acm.timus.ru. А именно, реализовать запрос "найти все узлы которые ниже и левее на координатной плоскости (x,y)" (count_lefter_and_lower).

Но этот алгоритм не проходит по времени по той простой причине, что декартово дерево может быть несбалансированным. Например, когда добавляются ключи (1,1), (2,2), (3,3), ... декартово дерево не ветвится и превращается в длинный "бамбук".

```
int
count_lefter_and_lower(node_t *root, dkey_t x, dkey_t y, int bounded) {
   if(root) {
      if(root->x <= x) {
         if( root->y <= y && bounded)  {
            return root->size;
         } else {
            return ( (root->y <= y) ? 1 : 0 ) +
            count_lefter_and_lower(root->r, x, y, bounded) +
            count_lefter_and_lower(root->l, x, y, 1);

         }
      } else {
         return count_lefter_and_lower(root->l, x, y, 0);
      }
   } else {
      return 0;
   }
}

node_t *root = 0;
int *res;

void count_res( node_t *node, int level ) {
   if(node) {
      res[ count_lefter_and_lower(root, node->x, node->y, 0) -1 ] += 1;
   }
}

int main(int argc, char *argv[])
{
   int n, i, x, y;

   node_t *left = 0;

   scanf("%d", &n);
   res = (int*) calloc( n , sizeof(int) );
   for(i = 0 ; i < n ; i++) {
      scanf("%d%d", &y, &x);
      left = dt_add_right( &root, left, x, y );
   }
   count_size(root);
   prefix_traverse(root, count_res, 0);

   for( i = 0 ; i < n ; i ++) {
      printf("%d\n", res[i]);
   }

   return 0;
}
```

-- ArtemVoroztsov - 02 Aug 2006

Copyright © 2003-2022 by the contributing authors.