**Алгоритм экспериментального исследования алгоритмов поиска**

**(лабораторная работа № 4)**

**Именованные константы**

KolRazm – число исследуемых размеров таблиц

MaxTbl – максимальный размер массива для таблицы (должен быть кратным KolRazm)

Step = MaxTbl **div** KolRazm – шаг изменения размера исследуемой таблицы

Beg = Step – начальный размер таблицы

MaxExp = 1000 – число экспериментов

KolAlg – число исследуемых алгоритмов

//*Обнуление результатов*

**for** i:=1 **to** KolAlg **do**

**for** j:=1 **to** KolRazm **do**

**begin**

Rez1[i,j].Bez:=0;

Rez1[i,j].Usp:=0;

**end**;

n:=Beg; Razm:=1; Root:=**nil**; AVL:=**nil**;

**while** n <= MaxTbl **do**

**begin**

//*заполнение таблиц данными*

**for** i:=1 **to** n **do** Tbl[i]:=2\*i; //*заполнение четными числами*

CrtDBP(Root,n); //*построить дерево поиска*

CrtAVL(AVL,n); //*построить АВЛ-дерево*

**for** NExp:=1 **to** MaxExp **do** //*проведение эксперимента*

**begin**

//*успешный поиск*

Randomize;

z:=(random(n)+1)\*2; //*ключ для успешного поиска (четный)*

//*выполняется поиск исследуемыми алгоритмами*

{1} RezPoisk:=Posled(z,n); //*последовательный*

Rez1[1,Razm].Usp:=Rez1[1,Razm].Usp+RezPoisk.Cnt;

{2} RezPoisk:=Binar(z,n); //*бинарный*

Rez1[2,Razm].Usp:=Rez1[2,Razm].Usp+RezPoisk.Cnt;

{3} RezPoisk:=Inter(z,n); //*интерполяционный*

Rez1[3,Razm].Usp:=Rez1[3,Razm].Usp+RezPoisk.Cnt;

{4} RezPoisk:=PoiskDBP(Root,z); //*простое ДБП*

Rez1[4,Razm].Usp:=Rez1[4,Razm].Usp+RezPoisk.Cnt;

{5} RezPoisk:=PoiskDBP(AVL,z); //*АВЛ-дерево*

Rez1[5,Razm].Usp:=Rez1[5,Razm].Usp+RezPoisk.Cnt;

//*безуспешный поиск*

Randomize;

z:=(random(n+1))\*2+1; //*ключ для безуспешного поиска (нечетный)*

//*выполняется поиск исследуемыми алгоритмами*

{1} RezPoisk:=Posled(z,n); //*последовательный*

Rez1[1,Razm].Bez:=Rez1[1,Razm].Bez+RezPoisk.Cnt;

{2} RezPoisk:=Binar(z,n); //*бинарный*

Rez1[2,Razm].Bez:=Rez1[2,Razm].Bez+RezPoisk.Cnt;

{3} RezPoisk:=Inter(z,n); //*интерполяционный*

Rez1[3,Razm].Bez:=Rez1[3,Razm].Bez+RezPoisk.Cnt;

{4} RezPoisk:=PoiskDBP(Root,z); //*простое ДБП*

Rez1[4,Razm].Bez:=Rez1[4,Razm].Bez+RezPoisk.Cnt;

{5} RezPoisk:=PoiskDBP(AVL,z); //*АВЛ-дерево*

Rez1[5,Razm].Bez:=Rez1[5,Razm].Bez+RezPoisk.Cnt;

**end**;

//*вычисление среднего числа сравнений*

**for** i:=1 **to** KolAlg **do**

**begin**

Rez1[i,Razm].Usp:=Round(Rez1[i,Razm].Usp/MaxExp);

Rez1[i,Razm].Bez:=Round(Rez1[i,Razm].Bez/MaxExp);

**end**;

//*Динамические деревья (АВЛ-дерево уже есть)*

**for** NExp:=1 **to** MaxExp **do** //*проведение эксперимента*

**begin**

Randomize;

z:=random(2\*n)+1; //*ключ запроса*

RezPoisk:=PoiskDBP(AVL,z);

**if** RezPoisk.Pntr <> **nil** **then** //*найдено, удалить*

DelNodeAVL(AVL,RezPoisk.Pntr,KRot)

**else** //*не найдено, добавить*

AddNodeAVL(AVL,z,KRot);

Rez1[6,Razm].Usp:=Rez1[6,Razm].Usp+KRot;

**if** KRot > Rez1[6,Razm].Bez **then** Rez1[6,Razm].Bez:=KRot;

**end**;

Rez1[6,Razm].Usp:=Round(Rez1[6,Razm].Usp/MaxExp);

n:=n+Step; //*увеличение размера исследуемой таблицы*

Razm:=Razm+1; //*номер столбца результатов*

**end**;

**procedure** CrtDBP(var T:TMyTree; n:integer); //*построить ДБП*

//*T – указатель на корень дерева*

**var**

TmpTbl:array[1..MaxTbl] of integer;

i,j,temp:integer;

**begin**

//*удаление старого ДБП*

DelTree(T);

**for** i:=1 **to** n **do** TmpTbl[i]:=Tbl[i]; //*промеж. табл. данных*

//*делаем случайную перестановку*

Randomize;

**for** i:=n **downto** 2 **do**

**begin**

temp:=TmpTbl[i];

j:=random(i-1)+1;

TmpTbl[i]:=TmpTbl[j];

TmpTbl[j]:=temp

**end**;

//*построение ДБП*

**for** i:=1 **to** n **do** AddNode(T,TmpTbl[i]);

**end**;

**Примечания**

Алгоритмы поиска реализованы как функции, возвращающие результат типа

TRezPoisk = **record** Poz, Cnt : integer; Pntr : TMyTree; **end**;

В процедурах DelNodeAVL(AVL,RezPoisk.Pntr,KRot), AddNodeAVL(AVL,z,KRot) выходной параметр KRot – число вращений.

Элемент таблицы результатов исследования (массив Rez1) имеет тип

TUspBez = **record** Usp, Bez : real; **end**;

В элементе таблицы результатов исследования (массив Rez1) для динамических таблиц в поле Usp сохраняется среднее число вращений, в поле Bez – максимальное число вращений.

random(n) + 1, добавляется единица, поскольку нулевого ключа нет (генерируемый диапазон 0..n).