**Последовательные таблицы.**

Будем рассматривать неотсортированные таблицы.

K - количество элементов в таблице

N - длина вектора представления элементов таблицы

Векторное представление:

type элемент = record key ... body ...;

таблица = array [1..N] of элемент

end

key=...

body=...

Время поиска K/2

Списковое представление:

type элемент = record key... body ...;

связь=элемент;

procedure вставить (var table:таблица; var ключ:key; тело:body)

begin

if последний>=N then write(‘нет места’) else begin

последний:=последний+1;

table[последний].key:=ключ;

table[последний].body:=тело;

end;

with table[последний] do

key:=ключ;

body:=тело;

end

end

Предполагаем, что длина ключа и тела одна и та же.

procedure изменить(var table:таблица; var последний:integer)

var i,j:integer;

begin

table[последний+1].key:=ключ;

i:=1;

while not (table[i].key=ключ) do {это условие хотя бы раз выполняется}

i:=i+1;

if i=последний+1 then write(‘нет записи с ‘,ключ) else table[i].тело:=тело

end

Операции вставить и изменить имеют сложность K/2, где К - количество элементов в таблице.

Procedure Исключить(var table:таблица; var последний:integer)

var i:integer

begin {найти такое i: table[i].ключ=ключ и удалить этот элемент из table}

i:=1; | процедура

table[последний+1].ключ:=ключ; |

while table[i].ключ<>ключ do i:=i+1{условие инвариантности цикла}| поиска

if i<=последний then begin

while i<>последний do begin

table[i]:=table[i+1];

i:=i+1

end;

последний:=последний-1;

end else write(‘Такого элемента нет’)

end.

Сложность: К/2 - поиск

К/2 - сдвиг

К/2+К/2=К, то есть сложность линейна

body=...

key=...

type ссылка=звено;

звено=record ключ:key;

тело:body;

связь:ссылка

end;

var таблица:ссылка;

procedure ВСТАВИТЬ(var таблица,последний:ссылка; ключ: key; тело:body)

var элемент:звено;

begin

new(элемент);

элемент.ключ:=ключ;

элемент.тело:=тело;

элемент.связь:=nil;

последний.связь:=элемент;

последний:=элемент;

if таблица=nil then таблица:=элемент else последний.связь:=элемент;

последний:=элемент

end

Сложность не зависит от длины таблицы

procedure изменить (var таблица, последний:ссылка; ключ:key; тело:body)

{найти таблица.ключ = ключ и заменить таблица.тело на тело}

var следующий:ссылка;

begin {поиск элемента с заданным ключом}

if таблица<> nil then begin

new(следующий);

следующий.ключ:=ключ:

следующий.связь:= nil;

последний.связь:=следующий;

следующий:=таблица;

end;

{поиск}

while следующий.ключ<> ключ do следующий:=следующий.связь;

if последний.связь<>следующий then следующий.тело:=тело

else write(‘элемент не найден’);

{нужно уничтожить сгенерированный элемент}

dispose(последний.связь)

end

Сложность К/2

procedure удалить(var таблица, последний: ссылка; ключ: key);

var текущий: ссылка;

{если элемент последний или первый, то рассмотрим отдельно, иначе сдвинем ссылку и освободим память}

if {таблица пуста} then write (‘Таблица пуста’) else

if {в таблице один элемент} then

if {единственный элемент есть искомый} then {сделать таблицу пустой}

else write(‘нет искомого элемента в таблице’)

else write (‘нет искомого элемента в таблице’)

else {поиск в таблице}

new(текущий);

текущий.ключ=ключ;

текущий.связь:=nil;

последний.связь:=текущий;

текущий:=таблица;

while текущий.ключ<> ключ do begin

предок:=текущий;

текущий:=текущий.связь

end

if {первый элемент - искомый} then begin

текущий:=таблица;

таблица:= текущий.связь;

dispose(текущий)

end

if {последний- искомый (текущий=последний)} then begin

последний:=предок;

последний.связь:=nil;

dispose(текущий);

dispose(текущий.связь)

end

else begin

предок.связь:=текущий.связь;

dispose(текущий);

dispose(последний.связь)

end

end

Сложность = сложности поиска по линейному списку К/2

Таблицу нужно формировать так, чтобы наиболее часто встречаемые ключи находились в начале списка. Зная частоту встреча7емости ключей и отсортировав таблицу можно улучшить эффективность.

Сортированные последовательные таблицы.

Типы ключа и тела далее просты.

procedure вставить(var таблица: table; var последний: integer; ключ: key; тело:body)

var i:integer;

begin

if последний = N then write(‘таблица заполнена’) else begin

i:=последний;

{считаем, что все ключи упорядоченны по возрастанию, то есть

I(Kj)=I(Kt)+1

(Kj, Kt)R и не s: (Kj, Ks)R (Ks, Kt)R}

while (i>=1) and (таблица[i].ключ>ключ) do begin

таблица[i+1].ключ:=таблица[i].ключ;

таблица[i+1].тело:=таблица[i].тело;

i:=i-1

end;

таблица[i].тело:=тело;

таблица[i].ключ:=ключ

end

end

Сложность операции вставки для отсортированных таблиц возросла.

Выводы:

1) основная сложность операций в таблице - поиск. Для данной - линейна.

2)векторное представление хорошо, когда операции удаления и вставки относительно редки, а, если же нет, то предпочтение нужно отдавать списковому представлению.

3) Для последовательных таблиц понижение сложности можно достичь за счет использования информации о встречаемости ключей. Операцию поиска можно сократить за счёт сокращения длины путей поиска.