Задание

Реализовать программу сортировки односвязного списка методом рекурсивного слияния. В качестве исходных данных использовать случайные числа и слова текстового файла. Измерить «грязное» время выполнения программы и трудоемкость по основным операциям для различных значений

Сущность и особенности алгоритма

Рекурсия естественным образом «всплывает» в сортировках, основанных на перераспределении элементов исходной последовательности: их **разделении** на части и обратном **слиянии.** В процессе разделения к полученным частям применяется тот же самый алгоритм до тех пор, пока не получатся части единичной (или нулевой) размерности, «сборка» (слияние) частей осуществляется в обратном порядке – после возвращения из рекурсивного вызова.

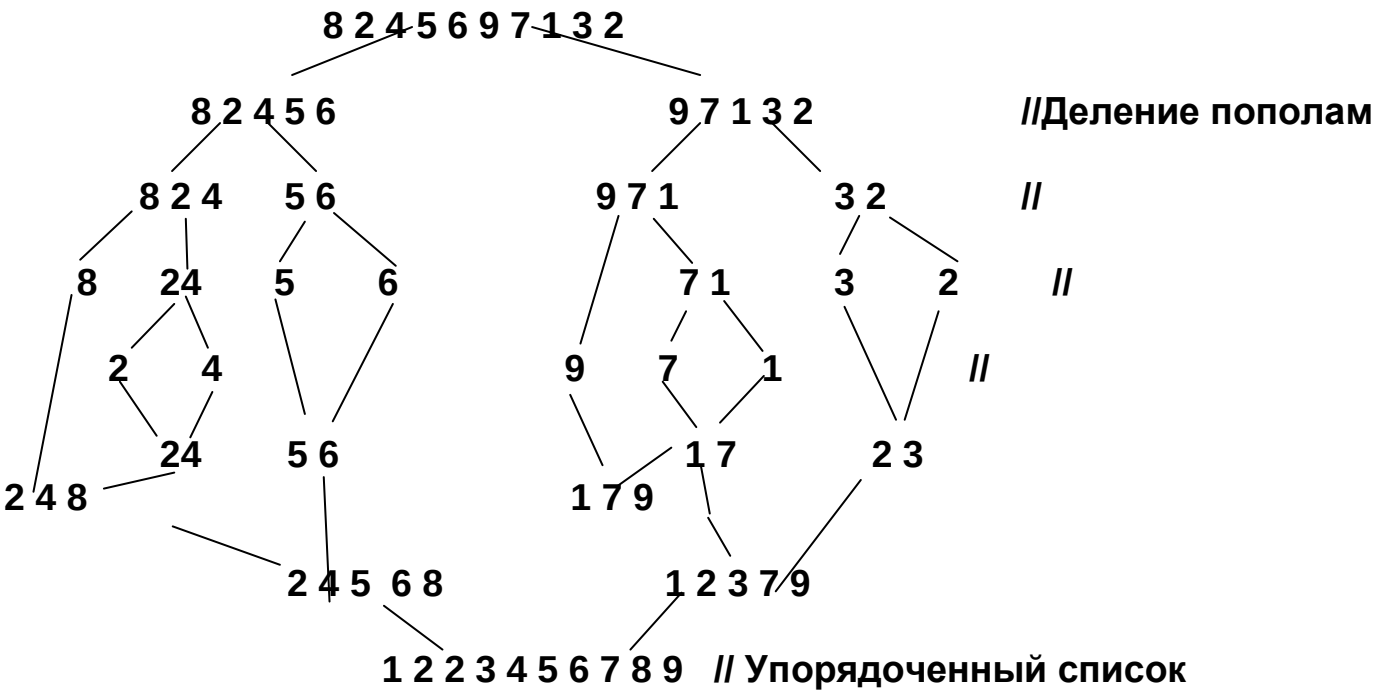
Сортировка рекурсивным слиянием. Если перенести «центр тяжести» на процесс слияния, то разделение последовательности станет простой формальностью: она просто «режется» на две части строго посередине. Это будет происходить до тех пор, пока не получатся части единичной или нулевой размерности. Содержательная сторона алгоритма будет заключаться в слиянии, которое будет выполняться в обратном порядке в процессе возвратов из рекурсивных функций (Если в тексте функции рекурсивного разделения вызов функции находится в конце текста, то в рекурсивном слиянии – в начале). Сама процедура слияния описана в 4.6: для получения одной упорядоченной последовательности из двух достаточно одного цикла, на каждом шаге которого выбирается (сливается) один элемент из пары «верхних».

Рекурсивное слияние односвязного списка. Рекурсивное слияние списка выглядит не так изящно, поскольку «разрезание» списка на две части требует лишнего цикла его просмотра с использованием указателей, движущихся с «одинарной» и «двойной» скоростью.

Оценка трудоемкости

Сортировка рекурсивным слиянием ввиду равномерности разделяемых частей близка к идеалу $\mathbf{T=N\log_2 N}$, однако требует дополнительной памяти для сливаемых частей.

Пример работы алгоритма



Изменения, вносимые в программу для оценки трудоемкости

Для оценки трудоемкости программы на числовых данных внесены изменения:

- добавлены счетчики операций сравнения, обменов и рекурсивных вызовов;
- вызывается функция clock() для измерения времени выполнения программы;

```
#include <stdio.h>
#include <string.h>
#include <iostream.h>
#include <fstream.h>
#include <time.h>
#include <stdlib.h>
#include <windows.h>

int cnt1=0;             // СРАВНЕНИЯ
int cnt2=0;             // ДВИЖЕНИЕ ПО СПИСКУ
int cnt3=0;             // ВЫЗОВЫ

struct list{
        char *val;
        list *next;
};
void show(list *p) {
        for (; p!=NULL; p=p->next) {
                char cc[80];
                CharToOem(p->val,cc);
                puts(cc);
        }
}

//-------------------------------------------------------72-05.cpp
//  сортировка односвязного списка рекурсивным слиянием
list *sort(list *p){
cnt3++;                                         //Вызовы
list *out,*p1,*p2,*q;
if (p==NULL || p->next==NULL) // не более одного - конец разделения
return p;                                               // найти середину, p2 движется в 2 раза
p1=p2=p;                                                // "быстрее" p1

while (p2->next!=NULL && p2->next->next!=NULL)
        {
        cnt2++;                                         //Движения по списку
        p1=p1->next;                            // 1-2-3-4-5-6-7 ==> 1-2-3-4 || 5-6-7
        p2=p2->next;
        p2=p2->next;
        }
q=p1->next;                                             // разделение списка на две части
p1->next=NULL;
p=sort(p);                      // рекурсивная сортировка частей
q=sort(q);
list OUTX;                                              // для выходного списка - фиктивный
p1=&OUTX;                                               // "последний" элемент
while(p!=NULL || q!=NULL)               // пока оба списка не пусты
        {                                                               // второй пуст или в первом меньше, чем во втором
        if (q==NULL || p!=NULL && (cnt1++,strcmp(p->val,q->val)<0))  //Сравнения
                p1->next=p,                             // перенести из первый из первого списка
                p1=p1->next,                    // в конец выходного
                p=p->next;
        else
                p1->next=q,                             // перенести из первый из второго списка
                p1=p1->next,                    // в конец выходного
                q=q->next;
        }
return OUTX.next;}
/*
```

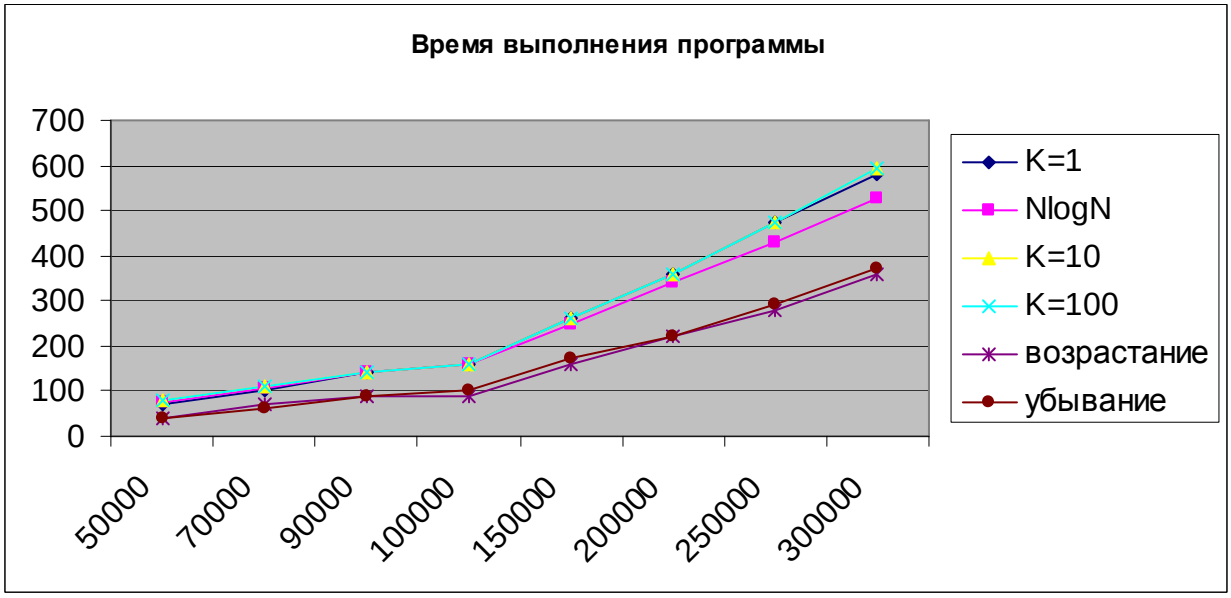
```
void main(){ list *ph=create(37);
show(ph); ph=sort(ph); show(ph);
}
*/
void main(){
int i,N,K=1;
int NN[]={5000,10000,20000,30000,40000,100000,150000,200000,0};
printf("N       time    cmp     moves   calls\n");
for (int kk=0;NN[kk]!=0;kk++){
        srand(time(NULL));
        N=NN[kk];
        cnt1=cnt2=cnt3=0;
   list *ph=NULL;
        ifstream F("1.txt");
        for (i=0;i<N;i++){
                if (!F.good()) break;
                char c[80];
                F >> c;
                list *q=new list;
                q->val=strdup(c);
                q->next=ph;
                ph=q;
                }
        N=i;
        //for (i=0;i<N;i++) a[i]=i;
        //for (i=0;i<N;i++) a[i]=N-i;
        long t=clock();
        ph=sort(ph);
        //show(ph);
        printf("%-10d%-10d%-10d%-10d%-10d\n",N,clock()-t,cnt1,cnt2,cnt3);
        while(ph!=NULL){
                list *q=ph;
                ph=ph->next;
                delete q;
                }
        }
```

Результаты измерений

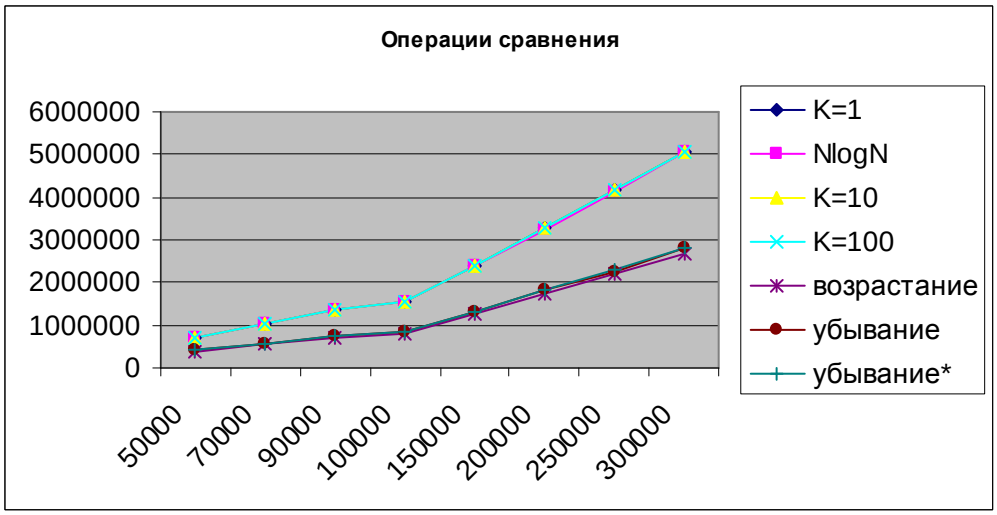
Данные – случайные числа

Грязное время

N	K=1	теория	отклонение	коэф по N1	K=10	K=100	возрастание	убывание
50000	70	75	-7%	0,0000963296	80	80	40	40
70000	100	109	-8%		110	110	70	60
90000	140	143	-2%		140	140	90	90
100000	160	160	0%		160	160	90	100
150000	260	248	5%		260	260	161	171
200000	361	339	6%		361	361	221	221
250000	472	432	9%		472	472	281	291
300000	582	526	11%		592	592	361	371



N	K=1	матем	отклон	коэф по N1	K=10	K=100	возрастание	убывание	матем
50000	718291	721893	0%	0,9249327235	718263	718263	382512	401952	401247
70000	1038978	1042080	0%		1039131	1039131	555200	573728	579215
90000	1369525	1369999	0%		1369458	1369458	723680	765248	761481
100000	1536280	1536280	0%		1536513	1536513	815024	853904	853904
150000	2392240	2385578	0%		2392064	2392064	1264224	1323632	1325966
200000	3273183	3257547	0%		3272429	3272429	1730048	1807808	1810628
250000	4168580	4146374	1%		4168629	4168629	2221248	2266608	2304661
300000	5084256	5048635	1%		5084861	5084861	2678448	2797264	2806161



Рекурсивные вызовы

N	K=1			K=10	K=100	возрастание	убывание
50000	99999			99999	99999	99999	99999
70000	139999			139999	139999	139999	139999
90000	179999			179999	179999	179999	179999
100000	199999			199999	199999	199999	199999
150000	299999			299999	299999	299999	299999
200000	399999			399999	399999	399999	399999
250000	499999			499999	499999	499999	499999
300000	599999			599999	599999	599999	599999
						2N-1	**2N-1**

Обмены

N	K=1	матем	отклонение	K=10	K=100	возрастание	убывание	K=1/K=100
50000	351953	354258	-1%	351953	351953	351953	351953	1,00
70000	503729	511384	-1%	503729	503729	503729	503729	1,00
90000	675249	672305	0%	675249	675249	675249	675249	1,00
100000	753905	753905	0%	753905	753905	753905	753905	1,00
150000	1173633	1170684	0%	1173633	1173633	1173633	1173633	1,00
200000	1607809	1598589	1%	1607809	1607809	1607809	1607809	1,00
250000	2016609	2034767	-1%	2016609	2016609	2016609	2016609	1,00
300000	2497265	2477537	1%	2497265	2497265	2497265	2497265	1,00

Анализ результатов:

1. Грязное время работы программы подчиняется закону $N\log_2 N$,с отклонением до 10%,увеличение повторений не меняют характеристик, на упорядоченных данных время уменьшается в два раза.
2. Количество операций сравнения соответствует формуле **$T_{срав}=0{,}924N\log_2 N$** с отклонением до 1%. %,увеличение повторений не меняют характеристик, на упорядоченных данных количество операций уменьшается в два раза
3. Количество операций обмена соответствует формуле **$T_{обм}=0.453N\log_2 N$** с отклонением до 1%, на упорядоченных данных трудоемкость остается той же самой.
4. Трудоемкость рекурсивных вызовов связана с прекращением рекурсии при первоначальном деление интервала пополам, при всех видах данных зависимость сводится к формуле **2N-1**

Данные – слова из текстового файла

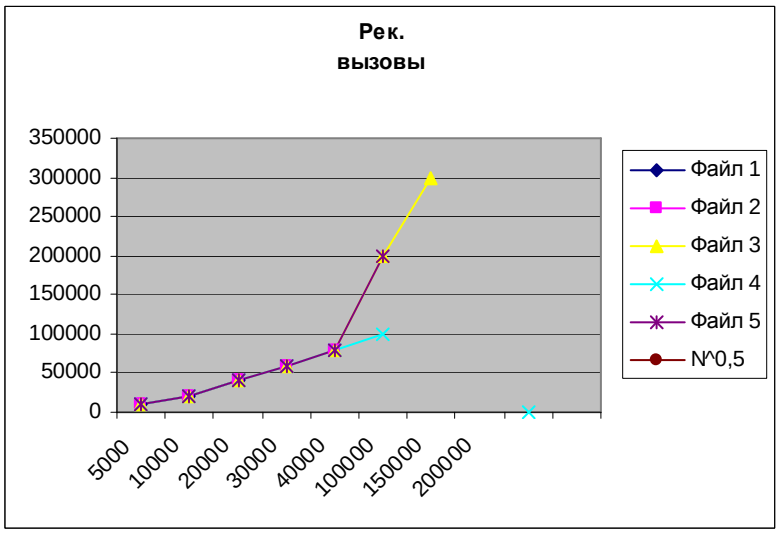
Производилось измерение производительности программы для 5 различных тестовых файлов. Оценка вида зависимости производилась по самому длинному файлу

	"Грязное" Время							
	Файл 1	Файл 2	Файл 3	Файл 4	Файл 5			
5000	10	10	10	10	0	11	-10%	Степень
10000	20	20	20	10	20	24	-17%	
20000	40	40	40	40	40	52	-23%	
30000		80	80	70	80	81	-1%	
40000		110	110	110	110	111	-1%	
100000			301	141	311	301	0%	0,0001812201
150000			502			467	7%	
200000								

	Сравнений							
	Файл 1	Файл 2	Файл 3	Файл 4	Файл 5	NlogN	Отклонение	
5000	55153	55173	55132	55176	55164	56804	-3%	
10000	120393	120321	120253	120234	120350	122854	-2%	
20000	260840	260640	260665	260619	260671	264200	-1%	
30000		408352	408461	408355	408431	412525	-1%	
40000		561401	561547	561527	561458	565383	-1%	
100000			1535679	717410	2E+06	1535679	0%	0,9245708854
150000			2391718			2384644	0%	
200000								

	Обмены							
	Файл 1	Файл 2	Файл 3	Файл 4	Файл 5		Отклонение	
5000	27005	27005	27005	27005	27005	27733	-3%	
10000	59009	59009	59009	59009	59009	59980	-2%	
20000	128017	128017	128017	128017	128017	128989	-1%	
30000		197729	197729	197729	197729	201404	-2%	
40000		276033	276033	276033	276033	276033	0%	0,4513968361
100000			753905	351953	753905	749754	1%	
150000			1173633			1164238	1%	
200000								

	Рек. Вызовы					
	Файл 1	Файл 2	Файл 3	Файл 4	Файл 5	
5000	9999	9999	9999	9999	9999	
10000	19999	19999	19999	19999	19999	
20000	39999	39999	39999	39999	39999	
30000		59999	59999	59999	59999	
40000		79999	79999	79999	79999	
100000			199999	99999	199999	
150000			299999			
200000						
				2N-1		



Анализ результатов:

1. Операции сравнения и обменов довольно точно соответствуют зависимости вида **$T_{срав}=0.92N\log_2 N$** с отклонением **3%** на малых размерностях и **$T_{обм}=0.451\ N\log_2 N$** с отклонением **3%** на малых размерностях.
2. На больших размерностях более **40000** имеются нулевые отклонения от линейно логарифмической зависимости.
3. «Грязное» время работы программы соответствует линейно логарифмической зависимости**.** При малых размерах файла вносится погрешность измерения времени в операционной системе (измеряется не время работы программы, а разность значений системных часов).

Выводы

1. На числовых данных и текстовых файлах в целом сортировка имеет линейно-логарифмическую трудоемкость и мало чувствительна к данным (в том числе повторению и упорядочению).