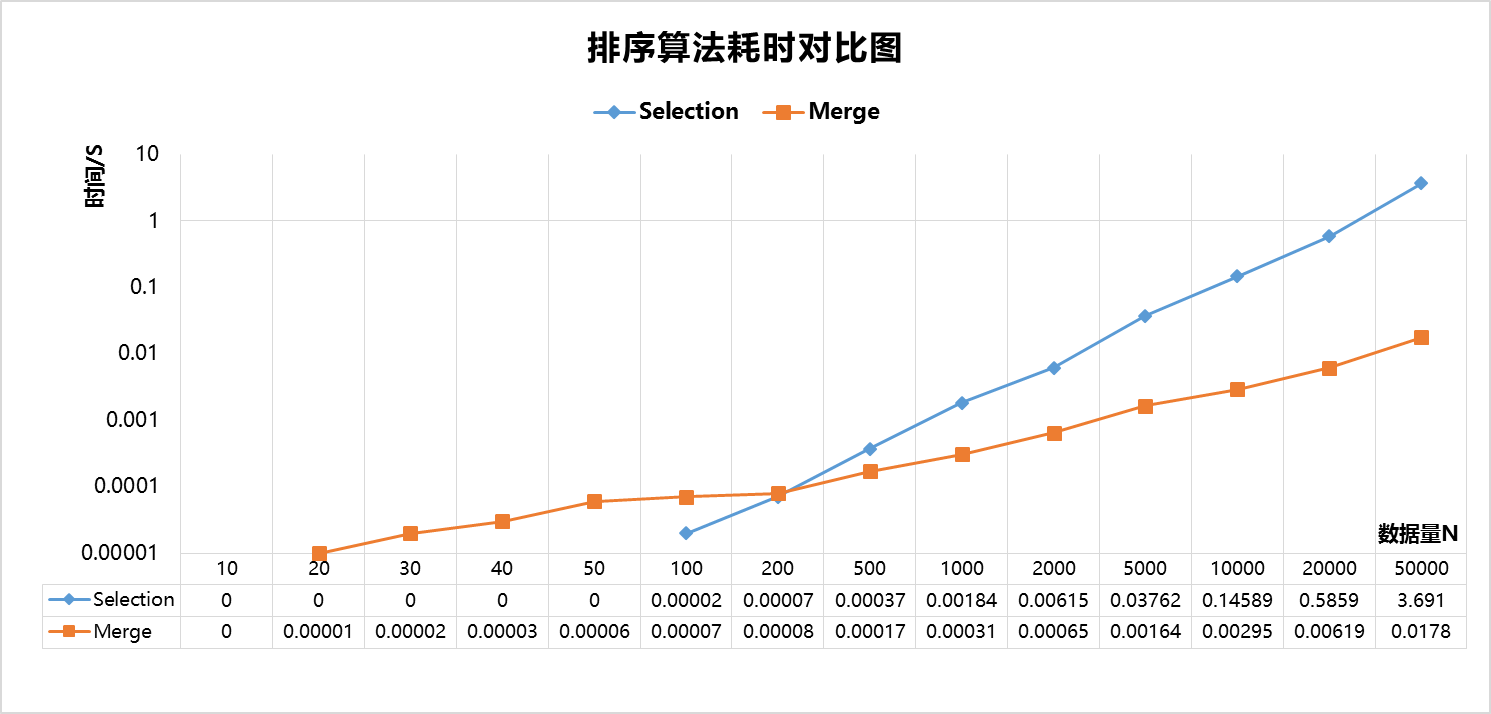
排序算法耗时对比

**张炽 2014010158 建管4**

对选择排序（Selection Sort）及归并排序（Merge Sort）两种排序算法在不同的数据规模下进行了算法完成时间的统计，其中每次实验重复100次，取平均时间，结果如下图所示。

上图表明，在数据规模N<200的情况下，选择排序算法要显著优于归并排序算法，耗时小于10的-5次方秒。当数据规模N>200后，选择排序算法的性能衰减显著增加，计算耗时随着数据量增加呈指数型增长，归并算法优势逐渐显现出来。在N=50000的数据量下，Merge算法是Selection算法速度的200多倍，在数据规模大的情况下，使用Merge算法能够大大提升排序速度，节省排序时间。

**附：题3 CODE**

#include<stdio.h>

#include <stdlib.h>

#include <time.h>

//生成随机数,范围从1-100

int (\*generator(int n)){

srand((unsigned) time(NULL));

int (\*A)[n];

int i;

A=calloc(n,sizeof(int));

for (i = 0; i < n; i ++){

(\*A)[i] = rand() % 100 + 1;

}

return A;

}

//定义merge

void merge(int \*A, int p, int r, int q){

int num1=r-p+1;

int num2=q-r;

int \*L1=(int \*)malloc(sizeof(int)\*(num1+1));

int \*L2=(int \*)malloc(sizeof(int)\*(num2+1));

int f;

for (f=0;f<num1;f++){

L1[f]=A[p+f];

}

L1[f]=100000;

for (f=0;f<num2;f++){

L2[f]=A[r+1+f];

}

L2[f]=100000;

int m=0;

int n=0;

int k;

for (k=0;k<q-p+1;k++){

if(L1[m]<=L2[n]){

A[p+k]=L1[m];

m++;

}

else{

A[p+k]=L2[n];

n++;

}

}

free(L1);

free(L2);

}

// 定义mergesort

void mergesort(int \*A, int p, int q){

if (p<q){

int r=(p+q)/2;

mergesort(A,p,r);

mergesort(A,r+1,q);

merge(A,p,r,q);

}

}

//定义selectionsort

void selectionsort(int \*A,int n){

int j,k,minj,minx;

for (k=0;k<=n-1;k++){

minj=k;

minx=A[k];

for (j=k+1;j<=n-1;j++){

if (A[j]<minx){

minx=A[j];

minj=j;

}

}

A[minj]=A[k];

A[k]=minx;

}

}

int main(){

int n=50000;

int (\*b)[n];

int repeat, repeat\_times = 100;

b=generator(n);

clock\_t begin = clock();

for (repeat=0;repeat<repeat\_times;repeat++){

selectionsort(b,n);

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC/repeat\_times;

printf("It took %f seconds to run selectionsort \n", time\_spent);

clock\_t begin1 = clock();

for (repeat=0;repeat<repeat\_times;repeat++){

mergesort(b,0,n-1);

}

clock\_t end1 = clock();

double time\_spent1 = (double)(end1 - begin1) / CLOCKS\_PER\_SEC/repeat\_times;

printf("It took %f seconds to run mergesort \n", time\_spent1);

return 0;

}