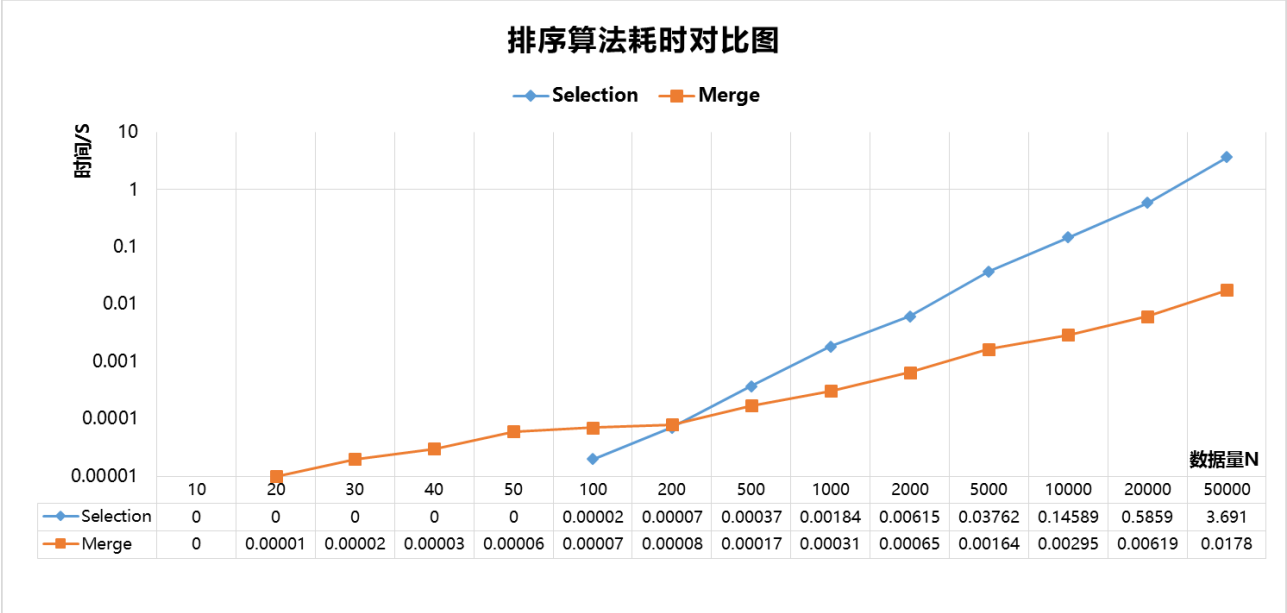
排序算法耗时对比

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。



	10	20	30	40	50	100	200	500	1000	2000	5000	10000	20000	50000
Selection	0	0	0	0	0	0.00002	0.00007	0.00037	0.00184	0.00615	0.03762	0.14589	0.5859	3.691
Merge	0	0.00001	0.00002	0.00003	0.00006	0.00007	0.00008	0.00017	0.00031	0.00065	0.00164	0.00295	0.00619	0.0178

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;
}
```