Data Structures
Active Learning – Internal Sorting

Younhyun Jung

Spring 2022

Problem

- Program two sorting algorithms
  - 1. Bubble Sorting - O(n^2), where n is the number of keys
  - 2. Any sorting algorithm that outperforms bubble sorting. You can google many sorting algorithms that our lecture did not cover

- Given a test sequence of 100000 keys, which is unsorted, produce a result using your sorting algorithm (as shown in below)
  - A template code will be given with two test sequences
  - A result from Bubble Sorting

```
Total time cost(ms) : 26027.000000
=============== RESULT ===============
Your sorting algorithm successfuly resulted in the correct ascending order for the given list
```

  - A result from Quick Sorting

```
Total time cost(ms) : 12.000000
=============== RESULT ===============
Your sorting algorithm successfuly resulted in the correct ascending order for the given list
```

A Template Code with the Test Sequence

```
// program your bubble sorting algorithm
void bubbleSorting(int* original_list, int numofkeys)
{

}

// program your second sorting algorithm
void yourSecondSorting(int* original_list, int numofkeys)
{

}

void sorting(int* original_list, int numofkeys)
{
    // bubbleSorting(original_list, numofkeys);
    // yourSecondSorting(original_list, numofkeys);
    return ;
}


int main()
{

    // read the test sequence
    int numofkeys = 0;
    FILE* fs = fopen(filename_keylist, "r");
    if (fs == NULL)
    {
        printf("The test seqeunce file (%s) is not accessible\n", filename_keylist);
        return 0;
    }

    while (fscanf(fs, "%d", &keylist[numofkeys]) == 1)
    {
        numofkeys++;

    }

    fclose(fs);


    // begin sorting with the test sequence
    clock_t start, end;
    start = (double)clock();
    sorting(keylist, numofkeys);
    end = (double)clock();

    // compute the time of sorting
    float computationTime = (double)(end - start); // get the total time cost
    printf("Total time cost(ms) : %lf \n", computationTime);


    // verify with the validation sequence
    int verifiedResult = check_sorted(keylist);

    printf("=============== RESULT =============== \n");
    if (verifiedResult == numofkeys)
    {
        printf("Your sorting algorithm resulted in the correct ascending order for th
    }
    else
        printf("Your sorting algorithm failed to produce the correct ascending order


}
```

```
59858
388
86267
24662
87789
283
43919
65238
85223
6722
99007
65255
63676
3563
44160
72196
47747
67927
42063
89257
59415
15091
40664
10430
62296
40104
1107
66232
9398
98279
58101
47566
93639
32495
75304
45826
69762
9341
33437
40960
41770
79105
13182
84494
83155
74603
67032
67230
```

What you need to submit

- Submit a pair of pseudo-code and C code for each of the two sorting algorithms
  - You can use Word for pseudo-code (see an example below)

```
Initialize n = Length of Array

BubbleSort(Array, n)
{
    for i = 0 to n - 2
    {
        for j = 0 to n - 2
        {
            if Array[j] > Array[j+1]
            {
                swap(Array[j], Array[j+1])
            }
        }
    }
}
```

  - You should fill your sorting algorithms into the template C code and submit your final code. Note that we won't accept any result that are not based on the template code.

Evaluation

- (50 points) if you submit your two sorting algorithms (Bubble sorting and another)
- (30 points) if your two sorting algorithms works correctly
  - We will use other five validation sequences (with 100000 keys)
- (20 points) 3 teams that produce low computations (times) using your second sorting algorithm (i.e., not Bubble Sorting), over all the groups
  - An ordinary PC will be used for the computation measures
  - We will average five repetitive trials