

# Running Time Survey

YAO ZHAO

# Running Time Survey

- ▶ This week, let's do a running time survey as a practice(**Practice 2**).
- ▶ You will be given a simple frame to do the running time survey of different algorithms on increasing inputs.

RunningTimeSurvey.java

# How to use?

- ▶ You should register your tasks and methods in the taskList

You can change the numbers based on your computer's performance.

```
public class RunningTimeSurvey {  
    //          task name          function name          run times upper  
    static String[][] taskList = {  
        { "LinearTimeTest",      "linearTime",      "10000000"},  
        { "LinearTimeTest",      "linearTimeCollections", "10000000"},  
        /*  
        * { "NlognTimeTest",      "NlognTime",      "1000000"},  
        * { "QuadraticTimeTest",  "QuadraticTime",  "100000"},  
        * { "CubicTimeTest",      "CubicTime",      "1000"},  
        * { "ExponentialTimeTest", "ExponentialTime", "29"},  
        * { "FactorialTimeTest",  "FactorialTime",  "12"},  
        */  
    };  
}
```

# LinearTimeTest

Since “linearTime” is registered for “LinearTimeTest”, you should define a function named linearTime, which looks like the following code:

```
public static long linearTime(int n) {  
    long[] list = new long[n];  
    generateList(n, list);  
    long timeStart = System.currentTimeMillis();  
    getMax(n, list);  
    long timeEnd = System.currentTimeMillis();  
    long timeCost = timeEnd - timeStart;  
    return timeCost;  
}
```

You can first write a function to generate data for your following algorithm.

Implements a Linear algorithm, for example, computing the maximum.

```
max ← a1  
for i = 2 to n {  
    if (ai > max)  
        max ← ai  
}
```

You can also choose other linear time algorithms.



# $O(n \log n)$ TimeTest

- ▶ You should register a new task named “NlognTimeTest”.
- ▶ You should register a function named “NlognTime”, the input parameter should be int, the return type should be long.
- ▶ You should generate your test data for your algorithm.
- ▶ You should implement your algorithm in which running time is required, for example, heap sort.

```
public static long NlognTime(int n) {  
    //TODO:generate you test input data here  
    long timeStart = System.currentTimeMillis();  
    //TODO: write a algorithm  
    long timeEnd = System.currentTimeMillis();  
    long timeCost = timeEnd - timeStart;  
    return timeCost;  
}
```

# QuadraticTimeTest

- ▶ Optional:
- ▶ Closest pair of points. Given a list of  $n$  points in the plane  $(x_1, y_1), \dots, (x_n, y_n)$ , find the pair that is closest.
- ▶  $O(n^2)$  solution. Try all pairs of points.

```
min ←  $(x_1 - x_2)^2 + (y_1 - y_2)^2$ 
for i = 1 to n {
    for j = i+1 to n {
        d ←  $(x_i - x_j)^2 + (y_i - y_j)^2$ 
        if (d < min)
            min ← d
    }
}
```

# CubicTimeTest

- Set disjointness. Given  $n$  sets  $S_1, \dots, S_n$  each of which is a subset of  $1, 2, \dots, n$ , is there some pair of these which are disjoint?  
 $O(n^3)$  solution: For each pair of sets, determine if they are disjoint.

```
foreach set  $S_i$  {  
    foreach other set  $S_j$  {  
        foreach element  $p$  of  $S_i$  {  
            determine whether  $p$  also belongs to  $S_j$   
        }  
        if (no element of  $S_i$  belongs to  $S_j$ )  
            report that  $S_i$  and  $S_j$  are disjoint  
    }  
}
```

# ExponentialTimeTest

- ▶ Given  $n$  bits, enumerate all possible Number.

$n$ -bit binary number

Example:  $n = 3$ ,  $\Rightarrow \{000, 001, 010, 011, 100, 101, 110, 111\}$



# FactorialTimeTest

- ▶ Brute force to compute factorial  $n$
- ▶ Use addition instead of multiplication

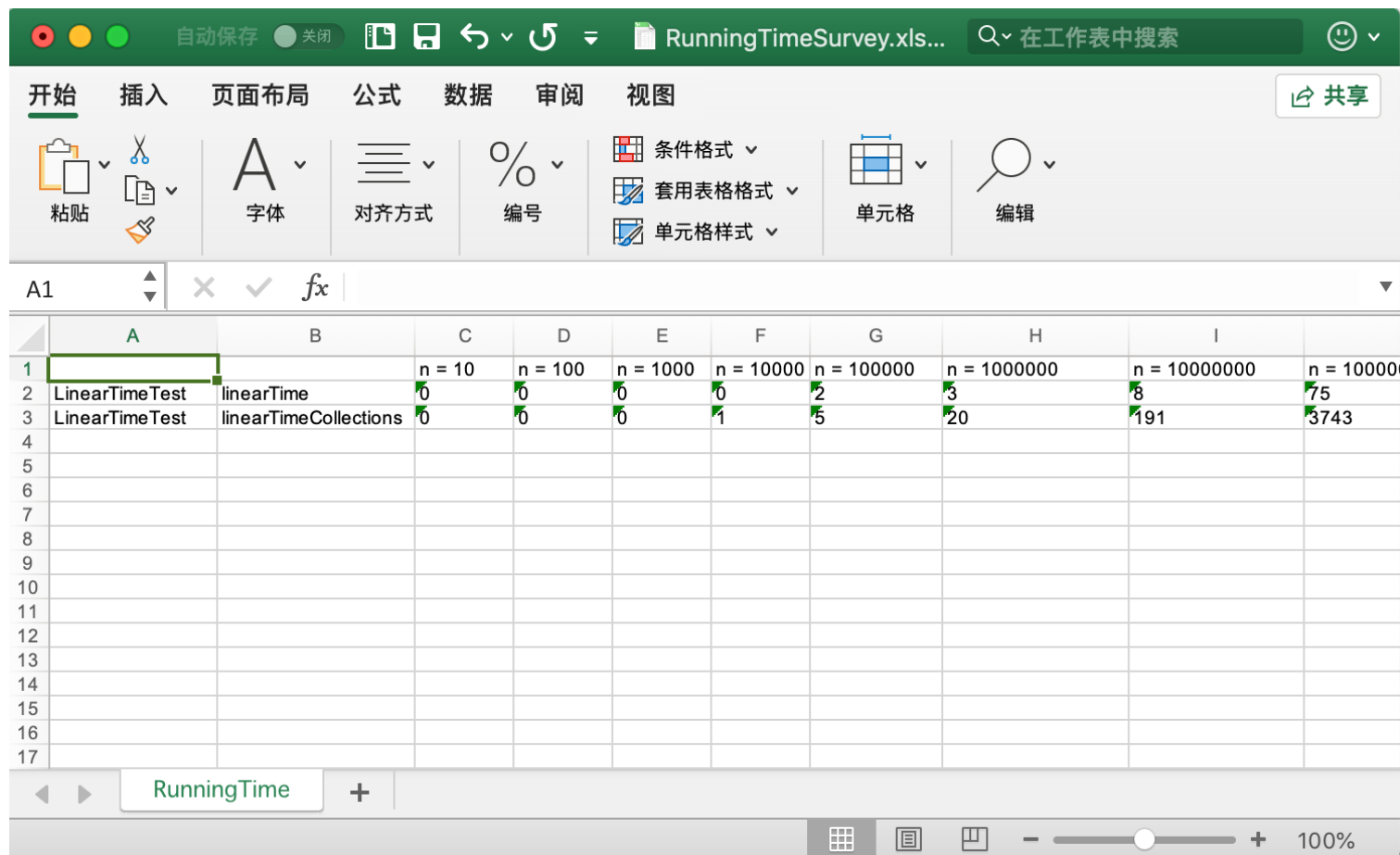
Generate all permutations of a list

# Optional: KPolynomialTimeTest

- ▶ Independent set of size  $k$ . Given a graph, are there  $k$  nodes such that no two are joined by an edge?
- ▶  $O(nk)$  solution. Enumerate all subsets of  $k$  nodes.

```
foreach subset S of k nodes {  
    check whether S is an independent set  
    if (S is an independent set)  
        report S is an independent set  
    }  
}
```

# Running result of the frame:



RunningTimeSurvey.xls...

在工作表中搜索

开始 插入 页面布局 公式 数据 审阅 视图

粘贴 字体 对齐方式 编号 条件格式 套用表格格式 单元格样式 单元格 编辑

A1

	A	B	C	D	E	F	G	H	I
1			n = 10	n = 100	n = 1000	n = 10000	n = 100000	n = 1000000	n = 10000000
2	LinearTimeTest	linearTime	0	0	0	0	2	3	75
3	LinearTimeTest	linearTimeCollections	0	0	0	1	5	20	191
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									

RunningTime

100%

100

[illegible]

The practice will be checked in this lab class or the next lab class (before **Mar.16**) by teachers or SAs.

This practice will contribute **1 mark** to your overall grade. Late submissions within 2 weeks after the deadline (before Mar.30) will incur a 20% penalty, meaning that you can only get 80% of the score.

Total score: 1 pt (nlogn, quadratic, cubic, exponential, factorial)  
Bonus: 0.1 pts