实验须知： 实验一（必做）；实验二、实验三（二选一）；实验四（必做）

实验一：快速排序

1．实验目的

1、熟悉并熟练掌握《算法导论》中所给出的快速排序算法。

2、发现某些极端情况的输入导致算法的不同表现。

3、体会在设计算法时要考虑到对极端情况的处理，体会“万年一遇的 bug”是如何产生的，体会算法的鲁棒性如何通过测试工程师的考验。

2. 实验学时

2 学时。

3. 实验问题

快速排序是算法导论中的经典算法。在本实验中，给定一个长为 n 的整数数组，要求将数组升序排序。

4. 实验步骤

4.1 按照算法导论中给出的伪代码实现快速排序

请严格按照如下伪代码实现算法，并在小样例上测试，保证实现的正确性。

```
QuickSort (A, p, r)
    if p < r
        q = Rand_Partition (A, p, r)
        QuickSort (A, p, q-1)
        QuickSort (A, q+1, r)
Rand_Partition (A, p, r)
    i = Random (p, r)
    exchange A[r] with A[i]
    x = A[r]
    i = p – 1
    for j = p to r – 1
        if A[j] <= x
        i = i + 1
        exchange A[i] with A[j]
    exchange A[i+1] with A[r]
    return i + 1
```

4.1 测试算法在不同输入下的表现

下面的各项测试中，“记录运行表现”的意思是，如果算法能够运行出结果，则记录算法的运行时间；如果算法无法得出结果，则记录无法得出结果的原因，比如“超过 1 小时未能得出结果”或者“Stack Overflow”等。

(1) 令输入为 1000000 个随机 32 位整数，记录算法的运行表现；

(2) 令输入为 1000000 个 1，记录算法的运行表现；

(3) 保持输入规模为 10000，先向数组中添加 x%比例的 1，再向数组中加入随机 32 位整数直到填满数组。分别令 x=50，60，70，80，90，100，记录算法的运行表现。

(4) 分别令输入规模为 1000，5000，重复第(3)步的实验。

(5) (选做) 调用你所用的程序设计语言中自带的快速排序算法，如 c++ STL 中的 sort 或 qsort，观察它们对上述输入有没有无法得出结果的情况出现。

4.1 思考并解决问题

按照算法导论中给出的伪代码实现的算法会在某些输入下无法得出结果。思考并回答：这种问题是如何产生的？如何解决？按照你的思路修改你的实现的快速排序算法，解决这一问题。

实验二：分治算法

1. 实验目的

1、掌握分治算法的设计思想与方法，

2、熟练使用高级编程语言实现分治算法，

3、通过对比简单算法以及不同的分治求解思想，体验分治算法。

2. 实验学时

6 学时。

3. 实验问题

求解凸包问题：输入是平面上 n 个点的集合 Q，凸包问题是要输出一个 Q 的凸包。其中，Q 的凸包是一个凸多边形 P，Q 中的点或者在 P 上或者在 P 中。（详

情请见课件）

4. 实验步骤

4.1 实现基于枚举方法的凸包求解算法

提示：考虑 Q 中的任意四个点 A、B、C、D，如果 A 处于 BCD 构成的三角形内部，那么 A 一定不属于凸包 P 的顶点集合。

4.2 实现基于 Graham-Scan 的凸包求解算法

提示：具体算法思想见课件。

4.3 实现基于分治思想的凸包求解算法

提示：具体算法思想见课件。

4.4 对比三种凸包求解算法

（1）实现随机生成正方形(0,0)-(0,100)-(100,100)-(100,0)内的点集合 Q 的算法；

（2）利用点集合生成算法自动生成大小不同数据集合，如点数大小分别为(1000，2000，3000…)的数据集合；

（3）对每个算法，针对不同大小的数据集合，运行算法并记录算法运行时间；

（4）对每个算法，绘制算法性能曲线，对比算法。

实验三：搜索算法

1. 实验目的

1、掌握搜索算法的基本设计思想与方法，

2、掌握 A*算法的设计思想与方法，

3、熟练使用高级编程语言实现搜索算法，

4、利用实验测试给出的搜索算法的正确性。

2. 实验学时

6 学时。

3. 实验问题

寻路问题。以图 1 为例，输入一个方格表示的地图，要求用 A*算法找到并

输出从起点（在方格中标示字母 S）到终点（在方格中标示字母 T）的代价最小的路径。有如下条件及要求：

（1）每一步都落在方格中，而不是横竖线的交叉点。

（2）灰色格子表示障碍，无法通行。

（3）在每个格子处，若无障碍，下一步可以达到八个相邻的格子，并且只可以到达无障碍的相邻格子。其中，向上、下、左、右四个方向移动的代价为 1，向四个斜角方向移动的代价为$\sqrt{2}$。

（4）在一些特殊格子上行走要花费额外的地形代价。比如，黄色格子代表沙漠，经过它的代价为 4；蓝色格子代表溪流，经过它的代价为 2；白色格子为普通地形，经过它的代价为 0。

（5）经过一条路径总的代价为移动代价+地形代价。其中移动代价是路径上所做的所有移动的代价的总和；地形代价为路径上除起点外所有格子的地形代价的总和。比如，在右下的示例中，路径$\mathrm{A}\to\mathrm{B}\to\mathrm{C}$的代价为$\sqrt{2}+1(\text{移动})+0(\text{地形})$，而路径$\mathrm{D}\to\mathrm{E}\to\mathrm{F}$的代价为$2(\text{移动})+6(\text{地形})$。

A		
	B	C
D	E	F

4. 实验步骤

4.1 实现 A*搜索算法

以图 1 为样例，实现 A*算法并测试输出。输出的路径应该与图 1 中以圆点标示的路径一致，或者略有不同但是有一样的代价。规则和条件如第 3 节所述。

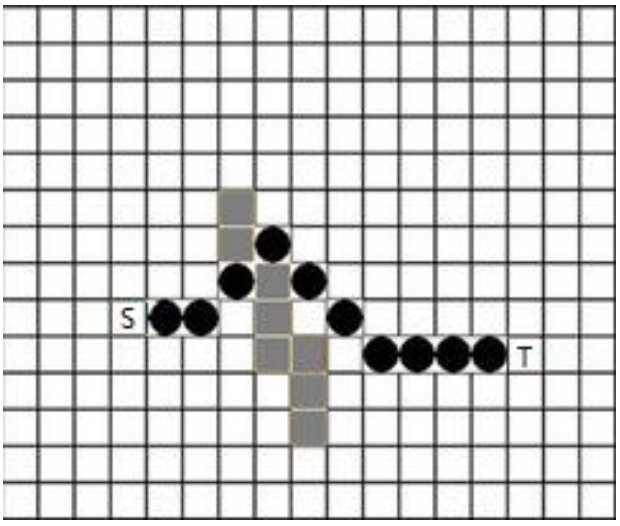

图 1

4.2 测试 A*算法的应用

精灵王子 Haposola 一大早从城堡中起床，要去沙漠中河流消失的地方寻找失落的宝藏。长路漫漫，处处险恶，精灵王子要尽快到达目标地点。

以图 2 为输入，寻找地图上给出的起点和终点间的最小代价路径。规则和条件如第 3 节所述。

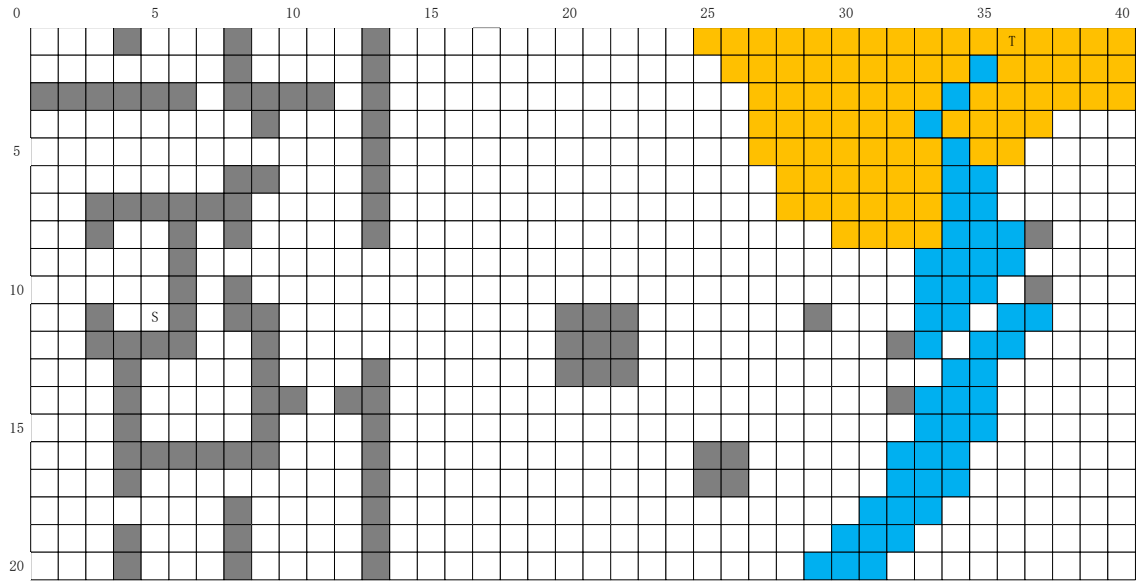

图 2

4.3 结果可视化

以适当的方式将地图可视化呈现，应该体现出方格、起点终点、障碍、地形等地图要素，并将算法计算出的最小代价路径在地图上体现出来。可以不完全遵守样例的形式（如颜色和以圆点表示的路径等），但以上要素应该完整地体现出来。

5. 考核须知

（1）需要展示 A*算法在两个输入下的输出。

（2）会根据同学们使用的可视化方法适当增减分数。

实验四：近似算法

1. 实验目的

1、掌握近似算法的基本设计思想与方法，

2、掌握集合覆盖问题近似算法的设计思想与方法，

3、熟练使用高级编程语言实现近似算法，

4、利用实验测试给出不同近似算法的性能以理解其优缺点。

2. 实验学时

8 学时。

3. 实验问题

集合覆盖问题：

- 输入: 有限集 X, X 的子集合族 F, $X=\cup_{S\in F} S$
- 输出: $C\subseteq F$, 满足

（1）$X=\cup_{S\in C} S$

（2）C 是满足条件(1)的最小集族, 即$|C|$最小.

4. 实验步骤

4.1 实现基于贪心策略的近似算法。

提示：具体算法思想见课件。

4.2 实现一个基于线性规划的近似算法。

提示：具体算法思想见课件。要求使用舍入法或随机舍入法。求解线性规划的步骤建议使用现成的库或工具，比如 C++的 GLPK 包，Python 的 pulp 包等。

4.3 测试算法性能

实验测试在|X|=|F|=100、1000、5000 情况。对于规模比较小的情况应该打印出可行解方案，在规模比较大的时候可以只给出运行时间。

注意：在问题规模为 5000 时算法很有可能运行非常长的时间。

数据生成建议:

（1）确保生成一组可行解集合$\{S_0,\cdots,S_m\}$，分别按照如下方法生成：

S_0：随机选 X 中的 20 个点放入其中；

S_1：产生一个[1,20]区间内的的随机数 n 代表S_1集合大小，再产生一个[1, n]区间内的随机数 x 代表需要从$X \setminus S_0$随机选 x 个点放入S_1中，另外从S_0中随机选$n-x$

个点放入S_1中；

S_{i+1}：依此类推，按照同样方法产生随机数 n 和 x，从$X \setminus \bigcup_{j=0}^{i} S_j$中随机选取 x 个点放入$S_{i+1}$中，从$\bigcup_{j=0}^{i} S_j$随机选$n-x$个点放入$S_{i+1}$中；

当$X \setminus \bigcup_{j=0}^{i} S_j$中的元素个数小于 20 的时候，直接作为最后一个集合S_m。

（2）生成其余集合

假设上述一共生成集合 y 个，自己再设计生成方法，生成其余|F|-y 个集合。

5. 考核须知

（1）注意实现的时候理解贪心近似算法设计思路。

（2）需要了解线性规划近似算法的设计思路，并通过实践理解其优缺点。

帮助

（1）记录算法运行时间方法(要求单位精确到毫秒)：利用各种高级语言提供的记录系统当前时间的函数，在程序中调用算法前记录系统当前时间，在程序中调用算法后记录系统时间，利用两个时间的差作为算法运行时间。

（2）绘制算法性能曲线，可以利用 Excel 软件的“插入折线图”等类似功能。算法性能曲线应该反映当数据集合从小变大时，具体算法的运行时间。