《大数据综合处理》课程设计
金庸的江湖

123456789 赵志辉*[1], 123456790 张三[1], 123456791 李四[1], and 233233233 喵喵喵[2]

[1] 计算机科学与技术系, 九乡河文理学院
[2] 喵喵喵, B612 星球

2020 年 12 月 9 日

1 插图

1.1 一张插图

图 1.1是一幅插图。

图 1.1: 一张截图

1.2 并排摆放的插图

图 1.2是 2 × 2 摆放的四张图片。

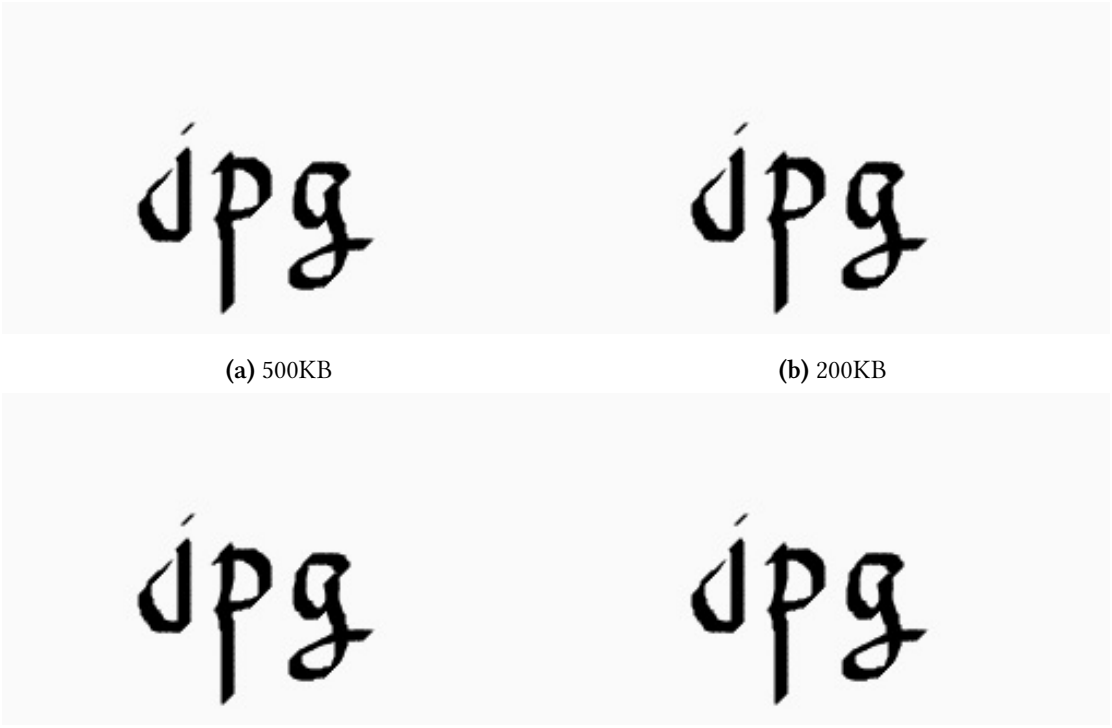
表 1.1是一个 subtable 的示例, 得到表 1.1a和表 1.1b。

1.3 环绕式插图

DDS—11A(T) 型电导率仪（附 DIS 型铂黑电极）1 台；计时器 1 只；恒温槽 1 套；双管式电导池 2 只（见图 1.3）；胖肚移液管（25mL）3 支；烧杯（50mL）1 只；容量瓶（250mL）一只；称量瓶（Φ25mm×23mm）1 只。

图 1.3: 双管式电导池示意图

*zhh.zhao@outlook.com



(a) 500KB **(b)** 200KB

(c) 100KB **(d)** 50KB

图 1.2: 相机拍摄的 FLIF 格式的图片在不同位置截断后的效果

表 1.1: 修正后的 κ_t 数据记录表

(a) 298.2K

t/min	κ_t/μS cm^{-1}	$\frac{\kappa_0-\kappa_t}{t}$/μS cm^{-1} min^{-1}
6	1892	49.7
9	1793	44.1
12	1712	39.8
15	1648	36.1
20	1559	31.6
25	1492	27.9
30	1439	25.0
40	1361	20.7
50	1305	17.7
60	1260	15.5

(b) 308.2K

t/min	κ_t/μS cm^{-1}	$\frac{\kappa_0-\kappa_t}{t}$/μS cm^{-1} min^{-1}
4	1788	73.0
6	1692	64.7
8	1621	57.4
10	1560	52.0
12	1512	47.3
15	1455	41.7
18	1405	37.5
21	1368	33.9
24	1336	31.0
27	1312	28.4
30	1288	26.4

2 代码

2.1 作为浮动体

代码 2.1是作为浮动体的代码块：

2.2 不作为浮动体

代码 2.2是并非浮动体的代码块：

代码 2.2：复读机

```
int main() {}
```

2.3 行内代码

这是行内的代码片段：`wstring price = L"九磅十五便士"`。

2.4 插入文件作为代码

代码 2.3是使用文件作为代码块的内容：

代码 2.3：Engine.hpp 的内容

```
#include <iostream>

int main() {
    std::cout << "Hello, world." << std::endl;
    return 0;
}
```

代码 2.1：凑字数用的代码

```
int main() {}
```

3 公式

3.1 独立成行的公式

下面是一些数学式：

$$\begin{aligned} P(x) &= \int_0^x \beta e^{-\beta x}\,\mathrm{d}x \\ &= \int_0^{\beta x} e^{-t}\,\mathrm{d}t \\ &= -e^{-u}\Big|_0^{\beta x} \\ &= 1 - e^{-\beta x}。 \end{aligned}$$

$$\begin{aligned} \text{maximize} \quad & H = \boldsymbol{p}^T \boldsymbol{q} \\ \text{s.t.} \quad & \boldsymbol{p}^T \mathbf{1} - 1 = 0。 \end{aligned}$$

3.2 行内公式

考虑 $\mathbf{H}_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$。又 $\Pr(X \geq 10^3 + 2000 | X \geq 2000) = \Pr(X \geq 10^3)$，

所以 $I * F = \begin{bmatrix} 2 & -2 & 1 & -1 \\ 4 & -5 & 4 & -3 \\ 2 & -3 & 3 & -2 \end{bmatrix}$

本证明受这个网页启发。

4 算法

习题 13.1

问题. *修改 Floyd-Warshall 算法，使其在给出所有点对之间的最短距离的同时，分别给出所有点对之间的：*

1. **后继路由表**，定义为矩阵 *GO*。*GO[i][j] = k*，表示从 *i* 到 *j* 的下一跳为 *k*。
2. **前驱路由表**，定义为矩阵 *FROM*。*FROM[i][j] = k*，表示从 *i* 到 *j* 的最后一跳为 *k*。

第 1 问 注意到，在 Floyd-Warshall 算法的进行途中，矩阵 `D` 中始终存放着当前的子问题下的最短路径，最终结束时子问题的最短路径则变为原问题的最短路径。所以我们应当在 `D` 被更新的时候同时更新路由表。算法如算法1所示。

第 2 问 如果要求前驱路由表而不是后继，我们只需简单地把算法1的行2改成

```
GO[i][j] = i
```

再把行8改为

```
GO[i][j] = GO[k][j]
```

即可。这样，`GO` 给出的将是从 `i` 到 `j` 的最短路径的最后一跳。

算法 1: 给出后继路由表的 Floyd-Warshall 算法

输入: 图 G 的邻接矩阵 D, 空的矩阵 GO

输出: G 的后继路由表 GO

```
1 forall i, j 使得 D[i][j] ≠ ∞ do
2 |   GO[i][j] = j;                                          // 初始化路由表
3 for k = 0 to n do
4 |   for i = 0 to n do
5 |   |   for j = 0 to n do
6 |   |   |   if D[i][k] + D[k][j] < D[i][j] then
7 |   |   |   |   D[i][j] = D[i][k] + D[k][j];
8 |   |   |   |   GO[i][j] = GO[i][k];                  // 发现更短路径,更新路由表
```

5 其他

5.1 带圆圈的编号列表

① 喵

② 呜