//´

1 第一章 算法在计算中的作用

1.1 算法

1. **给出现实生活中需要排序的一个例子或者现实生活中需要计算凸壳的一个例子。**
   答:
   排序: 图书馆查找图书需要根据用户需求按书名. 凸壳: 计算 N 个像控点最大面积应该是凸壳.

2. **除速度外, 在真实环境中还可能使用那些其他有关效率的量度.**
   答:
   程序效率: 计算复杂度; 工程效率: 工程进度

3. **选择一种你以前已知的数据结构, 并讨论其优势和局限**
   答:
   链表: 插入, 删除方便; 但查找不方便. 需要一个个遍历.

4. **前面给出的最短路径与旅行商问题有哪些相似之处? 又有哪些不同.**
   答:
   1: 两者都是寻找最短路径。
   2. 最短路径只需要找到最短的联通即可, 而旅行商需要每个节点都要遍历一次.

5. **提供一个现实生活的问题, 其中只有最佳解才行, 然后提供一个问题, 其中近似最佳的一个解也足够好.**
   答:
   估算 H 单应, 或者拟合各种线.

1.2 作为一种技术的算法

1. **给出应用层需要算法内容的应用的一个例子, 并讨论涉及的算法的功能.**
   答:
   Hash 表需要根据给定 KEY 生成对应地址方便程序快速查找并定位 KEY 对应的内容.

2. **假设我们正比较插入排序与归并排序在相同机器上的实现, 对规模为 n 的输入, 插入排序运行 $8n^2$ 步, 而归并排序运行 $64nlgn$ 步, 问对那些 n 值, 插入排序优于归并排序?**
   答:
   问题等价解 $8n^2 < 64nlgn$ n 是整数, 用 $n = \{1 \to n\}$ 一个个代入解得:

   $n = 1$: $8 > 0$
   $n = 2$: $32 < 128$
   ....
   $n = 43$: $14762 < 14933$
   $n = 44$: $15488 > 15373$

   即 $2 \leq n \leq 43$ 的时候插入排序优于归并排序.

3. **n 的最小值为何值时, 运行时间为 $100n^2$ 的一个算法在相同机器上快于运行时间为 2^n 的另一个算法?**
   答:
   问题等价解 $100n^2 < 2^n$ n 是整数, 也是简单代入解得: 当 $n \geq 15$ 时满足

2 第二章 算法基础

2.1 插入排序

1. **以图 2-2 为模型, 说明 INSERTION-SORT 在数组 A=31,41,59,26,41,58 上的执行过程**
   答:

序号:	1	2	3	4	5	6
	31	41	59	26	41	58
	31	41	59	26	41	58
	26	31	41	59	41	58
	26	31	41	41	59	58
	26	31	41	41	58	59

2. **重新过程 INSERTION-SORT, 使之按非升序 (而不是非降序) 排序**

```
1  Insertion-Sort(A)
2  for j = 2 to A.length
3    key = A[j]
4    i = j-1
5    while i>0 and A[i]< key //将判断大于改为小于即可.
6      A[i+1] = A[i]
7      i = i - 1
8    A[i+1] = key
```

3. **考虑以下查找问题:**
   输入: n 个数的一个序列 A=<a1,a2,...an> 和一个值 v
   输出: 下标 i 使得 v=A[i] 或者当 v 不在 A 中出现时,v 为特殊值 NIL.
   写出线性查找的伪代码, 它扫描整个序列来查找 v, 使用一个循环不变式来证明你的算法是正确的. 确保你的循环不变试满足三条必要的性质.
   答:

```
1  FIND-V(A,v)
2  i = 0
3  while i < A.length and  A[i] != v
4       i += 1
5       if i < A.length
6          OUT i
7       else
8          OUT NIL
```

   正确性:
   初始化:
   i = 0, while 循环迭代前不变式成立,
   保持:
   i 从 0 递增到 A.length. 如果 A[i] = v, 则循环体结束.
   终止:
   退出循环后, 判断 i 是否在 A.length 里, 如果在, 则输出 i, 否则 i 为 NIL.

4. **考虑把两个 n 位二进制整数加起来的问题, 这两个整数分别存储在两个 n 元数组 A 和 B 中, 这两个整数的和应按二进制形式存储在一个 (n+1) 元数组 C 中, 请给出该问题的形式化描述, 并写出伪代码.**
   答:
   因为 C 为 n+1 位, 故可以在 C[n+1] 存放 A+B 需要进位的数. 将 C 初始化为 0,
   当 A+B+C 大于 2 时, 说明需要进位, 故 C[n+1] 被置为 1, 而当前位为 (A+B+C)%2;
   当 A+B+C 小于 2 时, 说明不需要进位, 当前位等于 A+B

```
1  SUM(A,B,C,n)
2  C[0 to n+1] = 0
3  for i = 0 to n
4   if A[i] + B[i] + C[i] <= 2   //需要进位
5      C[i] = (A[i] + B[i] + C[i])%2   //当前位取2的模
6      C[i+1] = 1;     //实现进位
7   else
8      C[i] = A[i] + B[i]; // 0 或 1
```

2.2 分析算法

1. **用 Θ 记号表示函数** $n^3 \div 1000 - 100n^2 - 100n + 3$
   答:
   $\Theta(n^3)$

2. **考虑排序存储在数组 A 中的 n 个数: 首先找出 A 中的最小元素并首先找出 A 中的最小元素并将其与 A[1] 中的元素进行交换. 接着, 找出 A 中的次最小元素并将其与 A[2] 中的元素进行交换. 对 A 中前 n-1 个元素按该方式继续. 该算法称为选择算法, 写出其伪代码. 该算法维持的循环不变式是什么? 为什么它只需要对前 n-1 个元素, 而不是对所有 n 个元素运行? 用 Θ 记号给出选择排序的最好情况与最坏情况运行时间.**
   答:

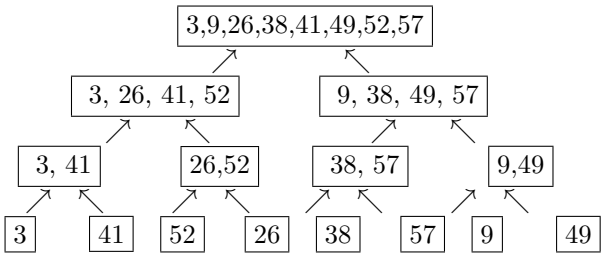
```
1  FUNC(A)
2  for i = 1 to A.length
3    min = i
4    for j = i + 1 to A.length //寻找min
5      if A[j] <= A[min]
6        min = j //更新min
7    A[i] = A[min]
```

3. **再次考虑线性查找问题 (参见练习 2.1-3. 假定要查找的元素等可能地为数组中的任意元素, 平均需要检查输入序列的多少元素? 最坏情况又如何? 用 Θ 记号给出线性查找的平均情况和最坏情况运行时间. 证明你的答案.**
   答:
   1. 平均需要检查输入序列的 n/2 个元素
   2. 最坏需要查找 n 个元素
   3. 平均情况和最坏情况运行时间为 Θ(n)

4. **应如何修改任何一个算法, 才能使之具有良好的最好情况运行时间?**
   答:
   针对最大概率的输入定制优化一个算法, 可获得最佳运行时间。

2.3 设计算法

1. **使用图 2-4 作为模型, 说明归并排序在数组 A={3,41,52,26,38,57,9,49} 上的操作;**
   答:



2. **重写过程 MERGE, 使之不使用哨兵, 而是一旦数组 L 或 R 的有元素均被复制回 A 就立刻停止, 然后把另一个数组的剩余部分复制回 A**
   答:

```
1  MERGE(A,p,q,r)
2  n1 = q - p + 1
3  n2 = r - q
4  let L[1 to n1+1] and R[1 to n2+1] be new arrays
5  for i = 1 to n1
6    L[i] = A[p+i-1]
7  for j = 1 to n2
8    R[j] = A[q+j]
9  i = 1; j = 1; k = p
10 while i <= n1 and j <= n2
11   if L[i] <= R[j]
12     A[k] = L[i]
13     i += 1
14   else
15     A[j] = R[j]
```

```
16      j += 1
17    k += 1
18  if i <= n1
19    while i <= n1
20      A[k] = L[i]
21      k += 1
22      i += 1
23  else
24    while j < n2
25      A[k] = R[j]
26      k += 1
27      i += 1
```

3. **使用数学归纳法证明: 当 n 刚好是 2 的幂时, 以下递归的解是 T(n)=nlgn.**

$$T(n) = \begin{cases} 2 & \text{若 } n = 2 \\ 2T(n/2) + n & \text{若 } n = 2^k, k > 1 \end{cases}$$

答:
n = 2 时, T(n) = nlgn = 2lg2 = 2, 原等式成立。
假定当 $n=2^k$,$k > 1$ 时:
T(n) = nlgn = $2^k \lg 2^k$ 成立.
则当 $n=2^{k+1}, k > 1$ 时:
T(n) = 2T(n/2) + n 将 $n=2^{k+1}$ 代入
= $2T(2^{k+1}/2)+2^{k+1}$
= $2T(2^k)+2^{k+1}$ 将 T(n)=nlgn 代入
= $2*2^k \lg 2^k+2^{k+1}$
= $2^{k+1}*k+2^{k+1}$
= $2^{k+1}*(k+1)$
= $2^{k+1}*\lg 2^{k+1}$
= nlgn

4. **我们可以把插入排序表示为如下的一个递归过程. 为了排序 A[1..n], 我们递归地排序 A[1..n-1], 然后把 A[n] 插入已排序的数组 A[1..n-1]. 为插入排序的这个递归版本的最坏情况运行时间写一个递归式.**
答:

```
1  BINARY-SEARCH-RECURSIVE(A, v, low, high)
2  if low < high
3       then return NIL
4  mid = (low + high)/2
5  if A[mid] = v
6       then return mid
7       elif A[mid] < v
8            then return BINARY-SEARCH-RECURSIVE(A, v, mid+1, high)
9            else return BINARY-SEARCH-RECURSIVE(A, v, low, mid-1)
```

5. **注意到 2.1 节中的过程 INSERTION-SORT 的第 5 7 行的 while 采用一种线性查找来 (反向) 扫描已排序好的子数组 A[1..j-1]. 我们可以使用二分查找 (参考练习 2.3-5) 来把插入排序的最坏情况总运行时间改进到 Θ(nlgn)?**
答:
因为需要移动元素的平均个数依然是 n/2, 因此最坏情况总运行时间还是 $\Theta(n^2)$.

3 第三章 函数的增长

3.1 渐近记号

1. **假设 $f(n)$ 和 $g(n)$ 都是渐近非负函数. 使用 Θ 记号的基本定义**
   证明 $max(f(n), g(n)) = \Theta(f(n) + g(n))$
   答:
   因为 $\Theta(f(n) + g(n))$ 为 $f(n) + g(n)$ 的上下界, 因此
   $0 \leq c1(f(n) + g(n)) \leq max(f(n) + g(n)) \leq c2(f(n) + g(n))$
   对所有 $n \geq n0$ 成立. 领 $c1 = \frac{1}{2}, c2 = 2$ 上式成立.

2. **证明: 对任意实常量 a 和 b, 其中 $b > 0$, 有 $(n + a)^b = \Theta(n^b)$**
   答:
   因为 a, b 是常数,$(n + a)^b$ 最高项为 $(Cn)^b$,C 为任意系数,Θ 由 n 最高次项决定, 即有 $(n + a)^b = \Theta(n^b)$

3. **解释为什么" 算法 A 的运行时间至少是 $O(n^2)$" 这一表述是无意义的.**
   答:
   O 记号表明算法运行时间渐近上界. 应该是说: 算法 A 的运行时间**至多**是 $O(n^2)$

4. $2^{n+1} = O(2^n)$ **成立么?** $2^(2n) = O(2^n)$ **成立么;**
   答:

3.2 标准记号与常用函数

1. **证明: 惹 f(n) 和 g(n) 是单调递增的函数, 则函数 f(n)+g(n) 和 f(g(n)) 也是单调递增的. 此外, 惹 f(n) 和 g(n) 是非负的, 则 f(n).g(n) 是单调递增的.**
   答: