**Advanced Algorithms**

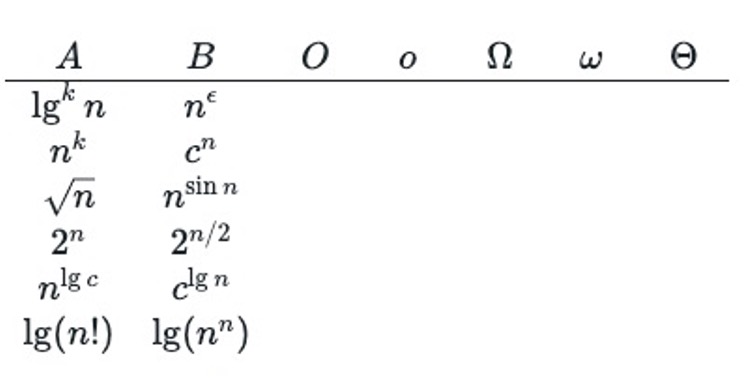
**Exercise for Lecture 2,3,4**

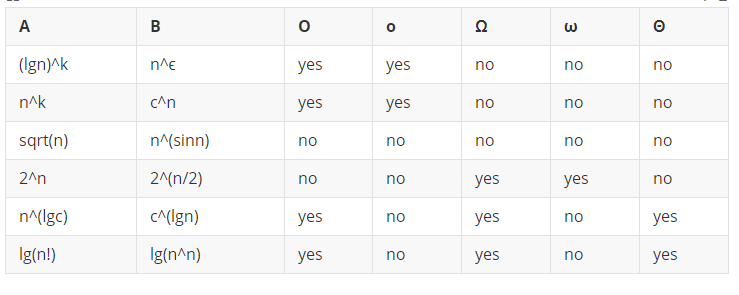
|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** | **周思源** | **Student ID** | 24S151092 |
| **Lecture 2** |  | | |
| **Lecture 3** |  | | |
| **Lecture 4** |  | | |
| **Total Score** |  | | |
| **Notes** | Deadline: **2024-09-22 24:00**  Submission Format: ‘**Lecture234\_Name\_ID.docx**’, and please send to:**aa\_24fall\_hw@163.com**  This assignment is meant to be an evaluation of your **individual** understanding coming into the course and should be completed **without collaboration** or outside help. | | |

**Lecture 2**

**Problem 2.1 [10 points]** Indicate, for each pair of expressions (A,B) in the table below, whether A is O, o, Ω, ω, or Θ of B. Assume that k≥1, ϵ>0, k≥1, ϵ>0, and c>1 are constants. Your answer should be in the form of the table with “yes” or “no” written in each box.

**Solution:**



****

**O上界Ω下界owθ同阶**

**Problem 2.2 [20 points]** Let f(n) and g(n) be asymptotically positive functions. Prove or disprove each of the following conjectures.

1. f(n)=O(g(n)) implies  g(n)=O(f(n)).
2. f(n)+g(n)=Θ(min(f(n), g(n))).
3. f(n)=O(g(n)) implies lg(f(n))=O(lg(g(n))), where lg(g(n))≥1 and f(n)≥1 for all sufficiently large n.
4. f(n)=O(g(n)) implies .
5. f(n)=O((f(n))2).
6. f(n)=O(g(n)) implies g(n)=Ω(f(n)).
7. f(n)=Θ(f(n/2)).
8. f(n)+o(f(n))=Θ(f(n)).

**Solution:**

1. 不正确，n = O(n^2)但n^2≠O(n)
2. 不正确，n^2 + n ≠ Θ(min(n^2, n)) = Θ(n)
3. 正确，由于f(n) = O(g(n))，存在c和n，使得n≥n0意味着f(n)≤c\*g(n)且f(n)≥1。这意味着log(f(n))≤log(cg(n)) = log(c) + log(g(n))。注意，取对数后不等式保留，因为f(n)≥1。现在我们需要找到d使得log(f(n))≤d\*log(g(n))使log(c) + log(g(n))≤d\*log(g(n))就足够了，这可以通过令d = log(c) + 1来实现，因为log(g(n))≥1。
4. 不正确，2n = O(n) 但 2^(2n) = 4^n ≠ O(2^n)
5. 正确，当f(n) >= 1时，0 <= f(n) <= c[f(n)]^2成立
6. 正确，利用4.2的对称性可知：想证明g(n) = Ω(f(n)), 只需要证明0 <= df(n) <= g(n), 令d = 1/c，不等式即成立
7. 不正确，令f(n) = 2^n即可证明不成立
8. 正确，令g(n) = o(f(n)), 则存在c > 0, n0 > 0有任意n > n0, 0 <= g(n) < cf(n); 只需证明存在c1 > 0, c2 > 0, n0 > 0: 任意n >= n0, 0 <= f(n) + g(n) <= c2f(n), 令c1 = 1, c2 = c + 1，上述不等式成立，得证；

**Lecture 3**

**Problem 3.1.[10 points]** Show that there is no comparison sort whose running time is linear for at least half of the n! inputs of length n. What about a fraction of 1/n of the inputs of length n? What about a fraction 1/?

**Solution:**

设决策树的高度为h，叶节点的个数为l

对于n!/2输入排列，n!/2<=l<=2^h，所以h>=lg(n!/2)=Ω(nlgn)

对于(n-1)!种输入排列，有(n-1)!<=l<=2^h，所以h>=lg((n - 1)!)=Ω(nlgn)

对于n!/2种输入排列，有n!/2^n<=l<=2^h，所以h>=lg(n!/2^n)=Ω(nlgn)

所以对于上述情况，都不存在一个比较次数为线性的排序算法

**Problem 3.2.[10 points]** Prove that COUNTING-SORT is stable.

**Solution:**

计数排序依靠首先统计不同元素的出现次数，并将这些次数存储在计数数组中，然后将计数数组转换为累计数组，这样可以知道每个元素在排序后输出中的位置，反向遍历输入数组后，根据累计计数将每个元素放在输出数组的位置上。

所以我们考虑两个数组中相等的数xi=xj，因为我们在最后输出的时候是反向遍历，所以假设i<j，此时xj总会在xi之前被放置在合理的位置，从而保证他们的相对顺序。

**Problem 3.3[10 points]** Show how to sort n integers in the range  0  to  −1  in O(n) time.

**Solution:**

可以是用基数排序来达到O(n)的目的；

因为数字的范围是从0-n^3-1，因此每个数字用基数n最多表示为3位数字，所以设其中一个数为x，则x=d2\*n^2+d1\*n+d0，其中d2,d1,d0为三个数位的值；

然后我们使用计数排序来进行基数排序：按照d0,d1,d2的顺序排序三个数位，最后得到答案；

计数排序的时间复杂度为O(n+k)，其中k为数字范围，k∈[0,n-1]

基数排序每一位的时间是O(n+n) = O(n)

因为基数有3位，所以基数排序会进行三次计数排序，总的时间复杂度为3\*O(n) = O(n)

**Problem 3.4 [10 points]** A probability distribution function P(x) for a random variable X is defined by  P(x)=Pr{X≤x}. Suppose that we draw a list of n random variables X1​, X2​,… Xn​ from a continuous probability distribution function P that is computable in O(1) time. Give an algorithm that sorts these numbers in linear average-case time.

**Solution:**

使用桶排序；

首先利用分布函数P(x)生成均匀分布的桶，概率分布函数可以将输入列表中的每个数字映射到区间[0,1]上的均匀分布，对于列表里的每个数字xi计算P(xi)。由于P(x)是一个连续的累积分布函数(CDF)，P(xi)将产生一个0-1内部的值。

然后将数字划分到桶里，创建n个桶，分别标记为B0…Bn-1，对于，每个数字xi，计算P(xi)并将其放入对应的桶中，因为分布是一个均匀分布，所以数字在每个桶的分布大致均匀。

对每个桶内的数字可以进行插入排序或者快速排序，因为每个桶中元素数量期望较少，大概为O(1)，所以对每个桶进行排序的时间平均为常数时间。

将所有桶按照顺序链接，就可以形成最终的排序链表，总的时间复杂度为O(n)

**Problem 4.1 [10 points]** Suppose we use a hash function *h* to hash *n* distinct keys into an array *T* of length *m*. Assuming simple uniform hashing, what is the expected number of collisions? More precisely, what is the expected cardinality of

**Solution:**

设关键字集合{k1,k2…kn},Pr{h(ki) = p} = 1/m

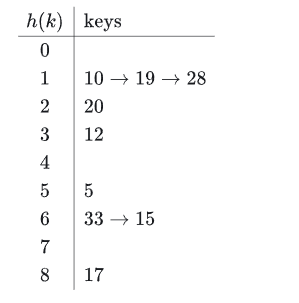
Pr{h(ki) = h(kj)} = ∑(0~m-1)Pr{h(ki) = p ∩ h(kj) = p}=

m\*(1/m)\*(1/m) = 1/m

所以对应的期望=∑(1~n)∑(i+1~n)Pr{h(ki) = h(kj)}=n(n-1)/2m

**Problem 4.2 [15 points]** Demonstrate what happens when we insert the keys 5,28,19,15,20,33,12,17,10 into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be *h*(*k*) = *k* mod 9.

**Solution:**



**Problem 4.3 [5 points]** Define a family  of hash functions from a finite set  to a finite set  to be ***ϵ-universal***  if for all pairs of distinct elements   and   in  ,

Pr{*h*(*k*)=*h*(*l*)}≤*ϵ*,

where the probability is over the choice of the hash function *h* drawn at random from the family . Show that an *ϵ*-universal family of hash functions must have

**Solution:**

令q=|Q|,u=|U|,设uj为槽位j∈Q中元素的个数，则u=∑(j∈Q)uj,对于给定的j，会发生冲突的次数为uj(uj-1)/2.

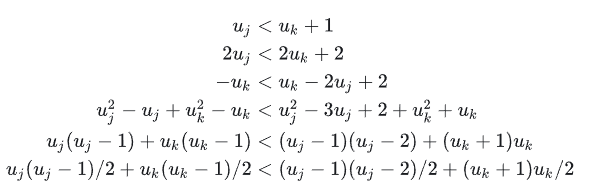
引理：

对于任一j∈Q，都有uj=u/q，则此时所有槽位发生冲突的次数总和最小。

证明：

如果uj<=u/q，我们称槽位j欠载，如果uj>u/q，我们称槽位j超载。

如果存在uj≠u/q这种不平衡的情况，先假设uj<=u/q，则必然存在另一个槽位k使得uk>=u/q，即uj<=u/q<=uk。两个槽位发生冲突的次数总和为uj(uj-1)/2+uk(uk-1)/2。



所以将欠载槽位的元素移到过载槽位只会增加发生冲突的次数总和，只有当所有元素均匀分布在各个槽位的时候发生冲突的次数总和最小。即对于任一个j∈Q，都有uj=u/q，则此时所有槽位的冲突次数总和最小，原命题得证。

此时，发生冲突的次数总和为q(u/q)(u/q-1)/2 = u(u/q-1)/2，不同元素对的个数为u(u-1)/2。

此时冲突发生概率最小：

