hw7

Q1

Question:

1. Recall that in the Knapsack Problem, we have n items, each with a weight w_i and a value v_i. We also have a weight bound W, and the problem is to select a set of items S of highest possible value subject to the condition that the total weight does not exceed W—that is, $\sum_{i \in S} w_i \leq W$. Here's one way to look at the approximation algorithm that we designed in this chapter. If we are told there exists a subset $\mathcal{O}$ whose total weight is $\sum_{i \in \mathcal{O}} w_i \leq W$ and whose total value is $\sum_{i \in \mathcal{O}} v_i = V$ for some V, then our approximation algorithm can find a set $\mathcal{A}$ with total weight $\sum_{i \in \mathcal{A}} w_i \leq W$ and total value at least $\sum_{i \in \mathcal{A}} v_i \geq V/(1+\epsilon)$. Thus the algorithm approximates the best value, while keeping the weights strictly under W. (Of course, returning the set $\mathcal{O}$ is always a valid solution, but since the problem is NP-hard, we don't expect to always be able to find $\mathcal{O}$ itself; the approximation bound of $1+\epsilon$ means that other sets $\mathcal{A}$, with slightly less value, can be valid answers as well.)

   Now, as is well known, you can always pack a little bit more for a trip just by "sitting on your suitcase"—in other words, by slightly overflowing the allowed weight limit. This too suggests a way of formalizing the approximation question for the Knapsack Problem, but it's the following, different, formalization.

Suppose, as before, that you're given n items with weights and values, as well as parameters W and V; and you're told that there is a subset $\mathcal{O}$ whose total weight is $\sum_{i \in \mathcal{O}} w_i \leq W$ and whose total value is $\sum_{i \in \mathcal{O}} v_i = V$ for some V. For a given fixed $\epsilon > 0$, design a polynomial-time algorithm that finds a subset of items $\mathcal{A}$ such that $\sum_{i \in \mathcal{A}} w_i \leq (1+\epsilon)W$ and $\sum_{i \in \mathcal{A}} v_i \geq V$. In other words, you want $\mathcal{A}$ to achieve at least as high a total value as the given bound V, but you're allowed to exceed the weight limit W by a factor of $1+\epsilon$.

Example. Suppose you're given four items, with weights and values as follows:

$$(w_1, v_1) = (5, 3), (w_2, v_2) = (4, 6)$$

$$(w_3, v_3) = (1, 4), (w_4, v_4) = (6, 11)$$

You're also given $W = 10$ and $V = 13$ (since, indeed, the subset consisting of the first three items has total weight at most 10 and has value 13). Finally, you're given $\epsilon = .1$. This means you need to find (via your approximation algorithm) a subset of weight at most $(1 + .1) * 10 = 11$ and value at least 13. One valid solution would be the subset consisting of the first and fourth items, with value $14 \geq 13$. (Note that this is a case where you're able to achieve a value strictly greater than V, since you're allowed to slightly overfill the knapsack.)

Answer：

首先对所有待选物品，如果$w_i > W$，将这件物品移除，不考虑，同时由于目前的$\sum_{i \in A} v_i = V$的选法是一定不包括这些要被移除的物品，因此可以移除不影响新的选法解和原问题的解的比较

对于剩下的n件物品，如果其$w_i \leq \frac{\epsilon W}{n}$, 则令$w_i' = 0$，对其他情况，令$w_i' = w_i - \frac{\epsilon W}{n}$

对于剩下的n件物品和新的二元组$< w_i', v_i >$，重新跑一遍0-1背包问题的动态规划算法要求总重量为W（是O（nV）的）

这样得到的最优解的总价值一定是$\geq V$的，同时假设这个解选中了至少k件物品$k \leq n$

总重量是$\sum w_i' = \sum w_i + k \times \frac{\epsilon W}{n} \leq \sum w_i + n \times \frac{\epsilon W}{n} = W(1+\epsilon)$

因此这是一个多项式时间的，满足条件的算法

Q2

Question：

Q2. （30 分）在一个包含 n 个元素的数组 A 中，$A[1] \neq A[n]$。我们希望找到一个索引 i，使 $A[i]$ 与 $A[i+1]$ 不相等。请考虑分治法，设计时间复杂度不超过 $\mathcal{O}(log(n))$ 的算法解决该问题。给出伪代码，并分析其运行时间。

Answer:

```
bineary_find(A, low, high)
    if low > high
        return -1
    mid = (high + low) / 2
    if A[mid] != A[mid + 1]
        return mid
    left = bineary_find(A, low, mid - 1)
    if left != -1
        return left
    return bineary_find(A, mid, high)
```

调用bineary_find(A, 1, n)

递归式为 $T(n) = T(n/2) + O(1)$

由主定理得，时间复杂度为 $T(n) = O(lgn)$

Q3

Question:

Q3.（30 分）小牛有一个朋友非常喜欢量子计算。ta 告诉小牛 Hadamard 门在量子算法设计中有非常重要的作用，并出了一道题考考小牛。

现在不考虑酉矩阵的性质，递归地定义 $2^n \times 2^n$ 矩阵 $\bar{H}_n$ 如下：

$$\bar{H}_0 = \begin{bmatrix} 1 \end{bmatrix},\ \bar{H}_n = \begin{bmatrix} \bar{H}_{n-1} & \bar{H}_{n-1} \\ \bar{H}_{n-1} & -\bar{H}_{n-1} \end{bmatrix}$$

。

给定一个 2^n 维的向量 v，求解 $\bar{H}_n v$。

请考虑分治法，帮助小牛设计一个时间复杂度不超过 $O(n2^n)$ 的算法进行求解。给出伪代码，并分析其运行时间。

Answer:

$$\begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} H_{n-1}V_1 + H_{n-1}V_2 \\ H_{n-1}V_1 - H_{n-1}V_2 \end{bmatrix}$$

```
function(matrix H_n, vector v)
    divide vector v into v1 and v2 each has 2^{n-1} dimension
    result1 = function(H_{n-1}, v1)
    result2 = function(H_{n-1}, v2)
    res1 = add(result1 , result2)
    res2 = sub(result1 , result2)
    let res1 and res2 to be a 2^n dimensions vector as res
    return res
```

递归式为$T(m) = 2T(m/2) + O(m)$，这里的$m = 2^n$

由主定理得 $T(m) = O(mlgm) = O(n2^n)$

Q4

Question:

Q4. (30 分) 虽然小牛做算法基础的作业题时常感到力不从心，但幸运的是，他有一群很友善的同学可以请教。由于同学们都有各自的事情要忙，只能简单告诉 ta 是否做对（是或者否）。提交了几次作业后，小牛发现有的同学判断相比其他同学会更准确。无疑，小牛想要寻找最能够帮助到 ta 的同学。可惜的是，他无法预先知道谁对他的帮助最大。为此，他考虑了这样一种策略：每做一道练习题，便去询问所有候选同学，采取“大多数权重”的意见：

1. 初始时，每个候选同学的权重均为 1；
2. 每次询问完后，采纳总权重更大的意见（是或者否）；
3. 在发现自己做的练习题是否正确后，将进行错误判断同学的权重减半。

(a) 请你告诉小牛使用这种策略至多会判断错误 $\frac{1}{2-\log_2 3}(m+\log_2 n)$ 次。其中 n 是候选同学的数目，m 是对小牛帮助最大的同学会判断错的数目。

(b) 使用随机策略的话，小牛可以做的更好。我们修改上述策略如下：

1. 按概率采纳一位候选同学的意见，其中概率与候选同学的权重成正比；
2. 对判断错误的候选同学的权重乘上因子 $0<\beta<1$。

请证明若小牛采取这种随机策略，判断错误的期望次数至多为

$$\frac{m\ln 1/\beta+\ln n}{1-\beta}$$

Answer:

(a)

由于每次判断都是选择总权重更大的意见，是第i次判断的总权重为w_i，则意见权重更大的那一方的总权重≥ $\frac{w_i}{2}$

如果判断错误，第i+1次判断的总权重≤ $\frac{w_i}{2}\times\frac{1}{2}+\frac{w_i}{2}=\frac{3w_i}{4}$

因此，每判断错误一次，总权重会缩小为原来的$\frac{3}{4}$

设当前每个人判断错误的次数为 x_i

则当前总权重为 $\sum_{i=1}^{n}2^{-x_i}$

经过若干次判断之后，除了帮助最大的同学判断正确m次以外，其他人无限次判断错误，这时候总权重$\sum_{i=1}^{n}2^{-x_i}=2^{-m}$

小牛判断错误的次数即 $log_{\frac{3}{4}}\frac{2^{-m}}{n}=\frac{1}{2-log_2 3}(m+log_2 n)$

此后小牛会选择帮助最大的同学，不会犯错误

(b)

假设在一次选择中，判断正确和错误的权重分别为a和b

这次选择，a有$\frac{b}{a+b}$的概率出错，权重变为$a+\beta b$

b有$\frac{a}{a+b}$的概率出错，权重变为$a\beta+b$

在选错的情况下，权重下降后比值的期望为

$$E=\frac{\frac{b}{(a+b)}\times(a+b\beta)+\frac{a}{a+b}\times(a\beta+b)}{a+b}=1+(\beta-1)\frac{\frac{a}{b}+\frac{b}{a}}{\frac{a}{b}+\frac{b}{a}+2}\leq e^{\beta-1}$$

由$n\times E^x=\beta^m$

$$x=log_E\frac{\beta^m}{n}=\frac{ln\frac{n}{\beta^m}}{lne^{1-\beta}}=\frac{-ln\beta+lnn}{1-\beta}$$