Jiun-Yan (Eric) Chen
CS 630 Homework #1
Collaborators: None

Please limit your answer to the following problems to at most 1/2 a page each.

Problem 1. You are given an integer t and a polynomial $p(x)$ of degree n where all of the $n+1$ coefficient a_i of p are non-zero integers. You want to compute $p(t)$.

i) How many integer additions and integer multiplications does it take to compute $p(t)$ the normal way, i.e. plugging in t for x wherever x occurs in $p(x)$ and then computing each term $a_i t^i$ from left to right and summing the values as you go?

To resolve each $a_i t^i$, there is a multiplication step $a_i \times t \times t \times ... \times t$ where there are i ts
So for each term, there is $i+1$ multiplication steps. As there is $i+1$ terms, the total multiplications count is:

$$\sum_{n=1}^{i+1} n = \frac{i+1(i+2)}{2}$$

To sum up all terms in $p(t)$, there is $a_i t^i + a_{i-1} t^{i-1} ... + a_0$, where there are i additions, as there are total of $i+1$ terms.

ii) Now see if you can find a better method to compute $p(t)$, using fewer than $\mathcal{O}(n^2)$ adds and multiplies. Describe your method and explain clearly how many arithmetic operations it uses.

There are two methods that could work, the first one makes the assumption that the algorithm can implement some form of caching.
If we could cache each t^n, then it would be a trivial $\mathcal{O}(1)$ operation to calculate t^{n+1}.
Thus the multiplications for each term $a_n t^n$ is $a_n \times t^{n-1} \times t$, which is 2 operations, and the total multiplication operations is $n \times 2$.

If we could not cache t^n, then we can also use a more efficient method of obtaining the powers, namely:

$$\begin{cases} t^n = t^{n/2} \times t^{n/2}, & \text{when n is an even number} \\ t^n = t^{(n-1)/2} \times t^{(n-1)/2} \times t, & \text{when n is an odd number} \end{cases}$$

With this method, we would have a minimum of $\log n$ operations, and a maximum of $2 \times \log n$ for each term. Summed up, this is equal to $n \log n$

For additions, as we will always have $n+1$ terms, we will always need to perform n addition operations.

Thus the total complexity for calculating a polynomial is $\mathcal{O}(2n+n) = \mathcal{O}(n)$ if caching is allowed, or $\mathcal{O}(n \log n + n) = \mathcal{O}(n \log n)$ if not.

Jiun-Yan (Eric) Chen
CS 630 Homework #1
Collaborators: None

Problem 2. A *permutation matrix* P is an $n \times n$ Boolean matrix with exactly one 1 in each row and one 1 in each column. It is called a permutation matrix because if you multiply P by any $n \times 1$ column vector v then the result Pv is a permutation of v.

i) Permutation matrices are invertible. Explain how to construct the inverse of a permutation P from P.

ii) What are the possible values of the determinant of P? Explain why your answer is true.

iii) Let $P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$. Construct the inverse of P and compute the determinant of P.

Jiun-Yan (Eric) Chen
CS 630 Homework #1
Collaborators: None

Problem 3.

i) Prove that the product of 2 lower triangular matrices is also lower triangular.

ii) Prove that the determinant of an upper triangular matrix is the product of its diagonal elements.

iii) Give an example of two 3×3 triangular matrices whose product is not triangular.

iv) Give an example of a non-singular 3×3 matrix M which has no LU decomposition, i.e. M is not equal to LU for an L and U where L is unit triangular and U is upper triangular.