Checkpoint 2

Shashank Singh
sss1@andrew.cmu.edu
1
February 2, 2012

Finding Well Spaced Triples of Ones

(a) The following pseudocode gives an algorithm which returns the triple `(j,i,k)` with the desired properties if such a triple exists, and returns `none` otherwise.

```
for i = 0,1,..,(n - 1)
    for j = 0,1,..,(i - 1)
        let k = 2*i - j
        if(k < n)
            if(a_j * a_i * a_k == 1)
                return (j,i,k)
return none
```

(b) For even $t \in \{0, 1, \ldots, 2n-2\}$, P_t is odd if and only if $a_{t/2} = 1$, and $P_t > 1$ if and only if, for some pair (j, k) with $|k - t/2| = |t/2 - j| \geq 1$, $a_j = a_k = 1$. Thus, P_t is odd and greater than 1 if and only if $(a_j, a_{t/2}, a_k)$ is a triple of ones with the desired properties, In particular, in this case, $\frac{P_t - 1}{2}$ gives the number of triples of ones centered around $a_{t/2}$. For odd t, P_t tells us nothing relevant to the problem at hand.

(c) Compute the convolution P of S with itself (this takes $O(n \log n)$ time using the discrete Fourier transform as discussed in lecture). Initialize a counter C at zero. For $t \in \{0, 2, 4, \ldots, 2n-2\}$, if $P_t \% 2 == 1$ and $P_t > 1$, then increment C by $\frac{t-1}{2}$. Then, return C.

(d) For $i \in \{0, 2, 4, \ldots, 2n-2\}$, if $P_i > 1$ and P_i is odd, iterate over $j \in \{i/2, \ldots, 1, 0\}$. If $a_j, a_{i/2}$, and a_{i-j} are all 1, then return the triple $(j, i/2, i-j)$. If the outer loop terminates without returning a triple, then no such triple exists, so return none. Note that, since $P_i > 1$ and then inner loop is will return before it goes through all of its iterations, so that the algorithm's runtime is $O(n)$.

Matrix Transposition In-Place

(a) Divide input matrix T of size $n \times n$ into four roughly square blocks T_1, T_2, T_3, T_4, where T_1 is the upper left quadrant of T, T_2 is the upper right quadrant, T_3 is the lower left quadrant of T, and T_4 is the lower right quadrant of T. Let C_1, C_2, C_3, C_4 be the corresponding quadrants of the T^T, the transpose of T. Then, $C_1 = T_1^T$, $C_2 = T_3^T$, $C_3 = T_2^T$, and $C_4 = T_4^T$, so that T^T can be computed recursively by computing the transposes of four matrices of size $\frac{n}{2} \times \frac{n}{2}$. Both dividing the input matrix into four submatrices and combining the four transposes of submatrices into the output matrix can be done in constant time. Thus, the runtime of this algorithm is given by the recurrence

$$T(1) = 1, T(n) = 4T(n/2) + k,$$

for some positive constant k. By the Master Method, $T(n) \in \Theta(n^2)$, as desired.

(b) Rather than modifying the structure of the matrix, whenever prompted for an entry in the matrix, say $a_{i,j}$, the entry in the j^{th} column of the i^{th} row of the matrix, return the corresponding entry in the transpose of the matrix, $a_{j,i}$. Thus, no work is done initially, and only constant time work is done when a value in the matrix is read.