THE UNIVERSITY OF
NEW SOUTH WALES

Algorithms: COMP3121/3821/9101/9801

Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

REVIEW PROBLEMS

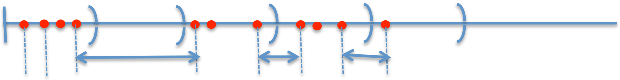
- Given n real numbers $x_1, ..., x_n$ where each x_i is a real number in the interval $[0, 1]$, devise an algorithm that runs in linear time and that will output a permutation of the n numbers, say $y_1, ..., y_n$, such that $\sum_{i=2}^{n} |y_i - y_{i-1}| < 1.01$.
- **Solution:** Split interval $[0, 1]$ into $100n$ equal buckets.
- Assign to each x_i its bucket number $b(x_i) = \lfloor 100n\, x_i \rfloor$;
- Why is this the bucket where x_i belongs?
- x_i is in the bucket k if and only if $\frac{k}{100n} \leq x_i < \frac{k+1}{100n}$;
- This holds if and only if $k \leq 100n\, x_i < k + 1$ i.e., if $k = \lfloor 100n\, x_i \rfloor$.
- So in linear time we can form pairs $\langle x_i, b(x_i) \rangle, \quad (1 \leq i \leq n)$;
- we can now sort these pairs according to their bucket number $b(x_i)$; since all bucket numbers are $< 100n$, CountingSort does that in linear time.
- Denote the sequence produced by CountingSort by $y_i, \quad 1 \leq i \leq n$.
- One can show that this sequence already satisfies the condition of the problem, but to make things simpler we do another extra step;

- We go through the sequence and in each bucket we find the smallest and the largest element;
- this can clearly be done in linear time;
- we now slightly change the ordering of each bucket: we always start with the smallest element in that bucket and finish with the largest element (leaving all other elements in the same order);
- we now prove that the sequence produced satisfies that $\sum_{i=2}^{n} |y_i - y_{i-1}| < 1.01$.
- we split this sum into two sums:

$$\begin{aligned}\sum_{i=2}^{n} |y_i - y_{i-1}| = &\sum_j \left(|y_j - y_{j-1}| : y_j \text{ and } y_{j-1} \text{ are in the same bucket}\right) + \\ &\sum_j \left(|y_j - y_{j-1}| : y_j \text{ and } y_{j-1} \text{ are in different buckets}\right)\end{aligned}$$

- we split this sum into two sums:

$$\sum_{i=2}^{n} |y_i - y_{i-1}| = \sum_j \big(|y_j - y_{j-1}| : y_j \text{ and } y_{j-1} \text{ are in the same bucket}\big) + \sum_j \big(|y_j - y_{j-1}| : y_j \text{ and } y_{j-1} \text{ are in different buckets}\big)$$



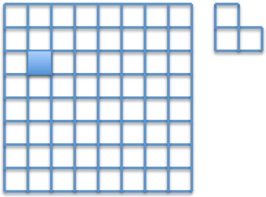
- note that there can be at most $n-1$ pairs of elements y_i, y_{i-1} which are in the same bucket (this happens if all elements end up in the same bucket!);
- whenever two elements y_i, y_{i-1} are in the same bucket $|y_i - y_{i-1}|$ is at most equal to the size of the bucket, i.e, $\leq \frac{1}{100n}$;
- thus, $\sum_j \big(|y_j - y_{j-1}| : y_j \text{ and } y_{j-1} \text{ are in the same bucket}\big) \leq \frac{n-1}{100n} < \frac{1}{100}$;
- $\sum_j \big(|y_j - y_{j-1}| : y_j \text{ and } y_{j-1} \text{ are in different buckets}\big) \leq 1$ because it is a sum of lengths of disjoint intervals in $[0,1]$;
- thus the total sum is smaller than $1 + 1/100 = 1.01$!

- Assume you are given two arrays A and B, each containing n distinct numbers and the equation $x^8 - x^4y^4 = y^6 + x^2y^2 + 10$. Design an algorithm which runs in time $O(n \log n)$ which finds if A contains a value for x and B contains a value for y that satisfy the equation.
- Solution: write the equation in the form

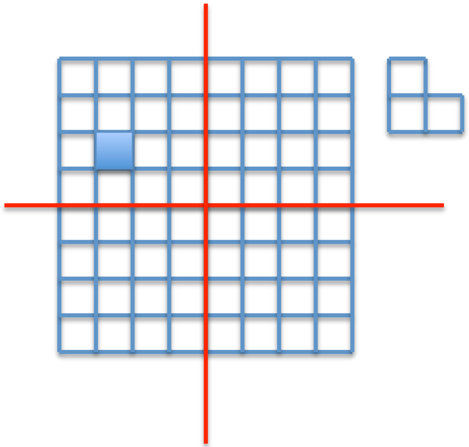
$$x^8 = x^4y^4 + y^6 + x^2y^2 + 10$$

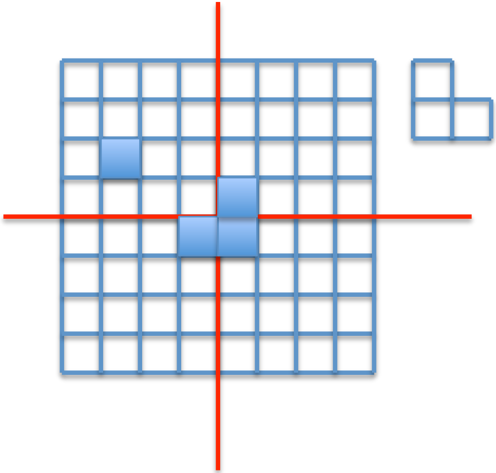
- now note that the right hand side is monotonic in y (actually, in both x and y, but monotonicity in y is important here.)
- sort array B in time $n \log n$;
- go through all elements of A; for each element $x \in A$ do binary search to see if there is a y satisfying the equation;
- this is possible because the right hand side is monotonic in y and B is sorted;
- if one element y produced a value of the right hand side bigger than the value of x^8 then all values larger than such y will also produce a value of the right hand side exceeding x^8.

- You are given a $2^n \times 2^n$ board with one of its cells missing (i.e., the board has a hole); the position of the missing cell can be arbitrary. You are also given a supply of "dominoes" each containing 3 such squares; see the figure:



Your task is to design an algorithm which covers the entire board with such "dominoes" except for the hole.





- Design an algorithm which multiplies a polynomial of degree 16 with a polynomial of degree 8 using only 25 multiplications in which both operands (which both depend on the coefficients of the polynomial) can be arbitrarily large.

- Multiply the following pair of polynomials using at most 7 multiplications of large numbers (large numbers are those which depend on the coefficients and thus can be arbitrarily large):

$$P(x) = a_0 + a_2x^2 + a_4x^4 + a_6x^6$$
$$Q(x) = b_0 + b_2x^2 + b_4x^4 + b_6x^6$$

- Multiply the following pair of polynomials using at most 3 multiplications of large numbers. (large numbers are those which depend on the coefficients and thus can be arbitrarily large):

$$P(x) = a_0 + a_{100}x^{100}$$
$$Q(x) = b_0 + b_{100}x^{100}$$

- Describe all k which satisfy $i\,\omega_{64}^{13}\omega_{32}^{11} = \omega_{64}^{k}$ (i is the imaginary unit).

- Compute all elements of the sequence $F(0), F(1), F(2), \ldots, F(2n)$ where

$$F(j) = \sum_{\substack{0 \leq i \leq n \\ 0 \leq j-i \leq n}} i^3 (j-i)^2$$

in time $O(n \log n)$.

Solution:

$$P(x) = \sum_{i=0}^{n} i^3 x^i; \qquad Q(x) = \sum_{j=0}^{n} j^2 x^j;$$

$$P(x)Q(x) = \sum_{j=0}^{2n} \left(\sum_{\substack{0 \leq i \leq n \\ 0 \leq j-i \leq n}} i^3 (j-i)^2 \right) x^j;$$

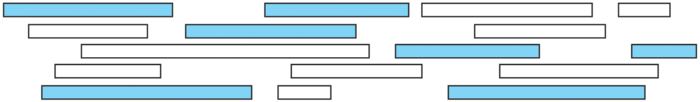
- So $F(j)'s$ are just the coefficients of the polynomial $P(x)Q(x)$.
- We can multiply two polynomials of degree n in $n \log n$ time using FFT.

- You are given a connected graph with weighted edges with all weights distinct. Prove that such a graph has a unique spanning tree.

- Assume that you are given a complete graph G with weighted edges such that all weights are distinct. We now obtain another complete weighted graph G' by replacing all weights $w(i,j)$ of edges $e(i,j)$ with new weights $w(i,j)^2$.

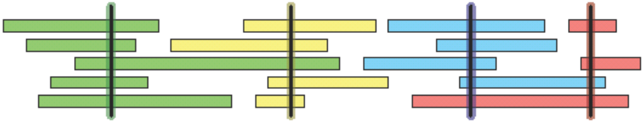
  Assume that T is the minimal spanning tree of G. Does T necessarily remain the minimal spanning tree for the new graph G'?

- Let X be a set of n intervals on the real line. A subset of intervals $Y \subseteq X$ is called a tiling path if the intervals in Y cover the intervals in X, that is, any real value that is contained in some interval in X is also contained in some interval in Y. The size of a tiling cover is just the number of intervals. Describe and analyse an algorithm to compute the smallest tiling path of X as quickly as possible. Assume that your input consists of two arrays $X_L[1..n]$ and $X_R[1..n]$, representing the left and right endpoints of the intervals in X.



A set of intervals. The seven shaded intervals form a tiling path.

- Let X be a set of n intervals on the real line. We say that a set P of points stabs X if every interval in X contains at least one point in P; see the figure below. Describe and analyse an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists of two arrays $X_L[1..n]$ and $X_R[1..n]$, representing the left and right endpoints of the intervals in X.


A set of intervals stabbed by four points (shown here as vertical segments)

- Assume denominations of your $n+1$ coins are $1, c, c^2, c^3, \ldots, c^n$ for some integer $c > 1$. Design a greedy algorithm which, given any amount, pays it with a minimal number of coins.

- Assume you have \$2, \$1, 50c, 20c, 10c and 5c coins to pay for your lunch. Design an algorithm that, given the amount that is a multiple of 5c, pays it with a minimal number of coins.

- Assume you are given n sorted arrays of different sizes. You are allowed to merge any two arrays into a single new sorted array and proceed in this manner until only one array is left. Design an algorithm that achieves this task and uses minimal total number of moves of elements of the arrays. Give an informal justification why your algorithm is optimal.

- Assume that you are given a complete weighted graph G with n vertices $v_1, \ldots, v_n$ and with the weights of all edges distinct. Assume that you are also given the minimal spanning tree T for G. You are also given an additional new vertex v_{n+1} and weights $w(n+1, j)$ of all new edges $e(n+1, j)$ between the new vertex v_{n+1} and all old vertices $v_j \in G$, $1 \leq j \leq n$. Design an algorithm which produces a minimum spanning tre T' for the new graph containing the additional vertex v_{n+1} and which runs in time $O(n)$.

- Along the long, straight road from Loololong to Goolagong there are n radio towers on a straight line. Each tower has an integer position and an integer radius of range. When a tower is activated, any towers within the radius of range of the tower will also activate, and those can cause other towers to activate and so on. We need an algorithm which finds the fewest number of towers you must activate to cause all towers to activate.

- What are the real and the imaginary parts of $e^{i\,\frac{\pi}{4}}$? Compute the DFT of the sequence $(1, 2, 3, 4, 5, 6, 7)$ by applying the FFT algorithm by hand.

Extende Classes Problems

- Assume that you are given an array A containing $2n$ numbers. The only operation that you can perform is make a query if element $A[i]$ is equal to element $A[j]$, $1 \leq i, j \leq 2n$. Your task is to determine if there is a number which appears in A at least n times using an algorithm which runs in linear time.

Extende Classes Problems

- Assume that you are given a complete binary tree with $2^n - 1$ many nodes, each node containing a distinct number. A node is a *local minimum* if the number it contains is smaller than the numbers contained in any of the nodes connected to it via an edge. Thus, the root would be a local minimum if its number is smaller than the numbers contained in both of its children; a leaf would be a local minimum if the number it contains is smaller than the number contained in its parent; any other number is a local minimum if the number it contains is smaller that numbers contained in its parent and both of its children. You can make queries by specifying a node; the query returns the value stored at that node. Design an algorithm which finds a local minimum (not all, just one) and makes only $O(n)$ queries.

Extende Classes Problems

- Assume that you are given an $n \times n$ table; each square of the table contains a distinct number. A square is a local minimum if the number it contains is smaller than the numbers contained in all neighbouring squares which share an edge with that square. Thus, each of the four corner squares has only two neighbours sharing an edge; $4(n-2)$ non corner squares along the edges of the table have three neighbouring squares and all internal squares have four neighbouring squares. A square is a local minimum if the number it contains is smaller than the numbers contained in all of its neighbouring squares. You can only make queries which given numbers m and k, $1 \leq m, k \leq n$, return the value in the square (m, k).
- Show by an example that if you search for a local minimum by picking a square and then moving to a neighbouring square with a smaller value (if there is such) might take $\Theta(n^2)$ many queries.
- Design an algorithm which finds a local minimum and makes only $O(n)$ many queries.

Extende Classes Problems

- Assume you are given two data bases A and B; A contains n numbers, B contains $n+1$ numbers, all numbers are distinct. You can query each database only in the following way: you specify a number k and the database (A or B); the query returns the value of the k^{th} smallest element in that database. Design an algorithm which produces the median of the set of all elements which belong to either A or B using at most $O(\log n)$ queries.