Quiz 1

- Do not open this quiz booklet until you are directed to do so. Read all the instructions first.
- The quiz contains 5 problems, several with multiple parts. You have 80 minutes to earn 80 points.
- This quiz booklet contains 10 pages, including this one, and a sheet of scratch paper.
- Write your solutions in the space provided. If you run out of space, continue your answer on the back of the same sheet and make a notation on the front of the sheet.
- Do not waste time deriving facts that we have studied. Just cite results from class.
- When we ask you to give an algorithm in this quiz, describe your algorithm in English or pseudocode, and provide a short argument for correctness and running time. You do not need to provide a diagram or example unless it helps make your explanation clearer.
- Do not spend too much time on any one problem. Generally, a problem's point value is an indication of how many minutes to spend on it.
- Good luck!

Problem	Title	Points	Parts	Grade	Initials
0	Name	1	11		
1	True/False	18	6		
2	Resizing van Emde Boas Trees	12	2		
3	Another Algorithm for MST	20	3		
4	Parity Problems	14	3		
5	K-Sum Subset	15	1		
Total		80			

Name: _____

Circle your recitation:

F10	F11	F11	F12	F1	F2	F3	F11	F12	F1
Kelly	Shalom	Victor	Angus	Ming	Devin	Paul	Brando	Nirvan	Daniel
R01	R02	R03	R04	R05	R06	R07	R08	R09	F10

Quiz 1-1: [18 points] T/F Questions

Mark each statement as either true or false. You have to **provide a short explanation for each**.

(a) Given an array, A, of n integers, we can output the set of elements of A with rank between $n/4$ and $3n/4$ in $O(n)$ time.

Solution. True; find the $n/4^{th}$ and $3n/4^{th}$ elements separately, then scan the list and output all elements that lie between them.

(b) It is possible to multiply two $n \times n$ matrices in time $o(n^3)$.

Solution. True by many possible arguments (e.g. cite lecture).

(c) Given a graph $G = (V, E)$ where $|E| = \Theta(|V|^2)$, Johnson's algorithm can be used to find all pairs shortest paths in time $o(V^3)$.

Solution. False. Johnson's only beats $O(V^3)$ time for sparse (i.e. $|E| = o(V^2)$ graphs.

(d) Let $G = (V, E)$ and let H be the subgraph of G induced by some set of vertices $V' \subset V$. That is, $H = (V', E')$ where E' consists of all edges both of whose endpoints are in V'. Then every MST of H is a subgraph of some MST of G.

Solution. False. The subgraph induced by two vertices is a single edge, and it is obvious that an arbitrary edge is not necessarily a subgraph of any MST of G.

(e) Suppose a Las Vegas probabilistic algorithm is run to termination on a set of inputs. You later find out that the psuedo-random number generator used by the algorithm to generate its randomness was corrupted, and instead the algorithm was using fixed numbers for randomness. Then the output of the Las Vegas algorithm is still guaranteed to be correct.

Solution. True. The output of a Las Vegas algorithm is always correct.

(f) If an algorithm runs in time $\Theta(n)$ with probability 0.9999 and in time $\Theta(n^2)$ with the remaining probability, then its expected run-time is $\Theta(n)$.

Solution. False.

Quiz 1-2: [12 points] Resizing van Emde Boas Trees

We modify the van Emde Boas data structure for storing elements in the range $\{1, \ldots, u\}$ so instead of having $\sqrt{u}$ blocks of size $\sqrt{u}$ at each node, we have 1000 blocks of size $u/1000$ each.

(a) [6 points] Write the recurrence for the run-time complexity of inserting an element to the new data structure. Solve the recurrence.

Solution.

The insert operation works much the same as before, except we recurse on blocks of size $u/1000$ instead of $\sqrt{u}$.

We have $T(u) = T(u/1000) + O(1)$. Thus $T(u) = O(\log u)$.

(b) [6 points] Write the recurrence for the space complexity of the new data structure and solve it.

Solution. We need an array of size 1000 to store the pointers to the recursive structures, and our summary bits can just be another size 1000 array instead of a recursive structure. (We actually don't even need the summary bits.) The space is then just the constant overhead of storing the max, min, and these arrays, plus the space needed to store the 1000 recursive structures.

So we have $S(u) = 1000T(u/1000) + O(1)$. With a simple recursion tree, or by case 2 of the Master Theorem, this solves to $S(u) = O(u)$.

Quiz 1-3: [20 points] Another Algorithm for MST

In this problem, you will develop another algorithm for finding the minimum spanning tree of a graph G.

- For each vertex v, choose the minimum weight edge adjacent to v and mark the edge.
- For each marked edge, contract the two vertices that it connects into one vertex.
- Repeat until there is only one vertex left.
- Return all marked edges.

(a) [5 points] Find the worst-case run time of the algorithm.

Solution. The run time is $O(|E| \log |V|)$. The reasoning is as follows:
We note that the number of vertices reduces at least by a factor of 2 at every iteration, so we will repeat the procedure at most $O(\log |V|)$ times. Determining the least weight edges for all the vertices can be done in $O(|E|)$ time by doing the following: at each node, keep a min value of the least weight adjacent edge and a pointer to the edge; then iterate over all edges, updating the min of the edge's endpoints as you go. Hence, the overall time is $O(|E| \log |V|)$.

(b) [5 points] Prove that at each iteration, all the marked edges belong to the MST of the input graph of that iteration. With this, briefly argue (in one sentence) that the algorithm returns the minimum spanning tree of G.

Solution. For an MST T, suppose (for the sake of contradiction) that there is a node v whose minimum weight edge e is not in T. Adding e to T will create a cycle. Since the cycle contains v, there must be another edge f adjacent to v in the cycle, and we know that $w(f) > w(e)$. Then the tree $T \setminus \{f\} \cup \{e\}$ is also a spanning tree, but with less weight than T, contradicting that T is an MST. This proves the first statement.
For the correctness of the algorithm, Let G' be the new graph after one iteration of contracting. We note that if T' is an MST of G' then $T' \cup \{\textit{edges marked in this iteration}\}$ is an MST of G. As a brief argument, we can simply induct on the number of iterations and the result follows. Below is a formal proof of the argument for interested readers.

Proof. Without loss of generality, suppose only one edge e is marked in this iteration. From (a), we know that e is in some MST T^* of G. Hence, $T^* \setminus \{e\}$ is a spanning tree of G'. Let T' be an MST of G', and hence we have

$$w(T') \leq w(T^* \setminus \{e\})$$

Adding $w(e)$ to both sides yields

$$w(T' \cup \{e\}) = w(T') + w(e) \leq w(T^* \setminus \{e\}) + w(e) = w(T^*)$$

Since T^* is an MST of G, this inequality implies that $T' \cup \{e\}$ is also an MST of G.
If there are more than one edge marked in this iteration, we can repeat the above argument and the result still holds.
As a final step, we induct on the number of iterations, and this completes the proof. □

(c) [10 points] A planar graph is a graph that can be drawn on a 2D plane (e.g., on paper) without any edges crossing. Edges are parallel if they connect the same two vertices. From graph theory, we know that for a planar graph $G = (V, E)$ with no parallel edges $|E| \leq 3|V|$.

Find an $O(|V|)$ time algorithm for finding the minimum spanning tree of a planar graph with no parallel edges. Modify the algorithm and analysis from previous parts.

Hint: If you contract two vertices in a planar graph, the resulting graph is still planar.

Solution. We modify the algorithm by removing parallel edges after contracting, leaving only the shortest edge between two nodes. The algorithm is still correct, for removing longer parallel edges does not change the MST.

As for the run time, the analysis above shows that there are at most $\log |V|$ iterations. We note that the graph stay planar after each iteration, and we remove parallel edges. Hence, if the original graph has $|E| \leq 3|V|$ edges, the graph after one iteration will have $\leq \frac{|V|}{2}$ vertices, meaning that it has $\leq 3\frac{|V|}{2}$ edges. Since the run time of each iteration is linear in the number of edges, the total run time would be $O(3|V| + 3\frac{|V|}{2} + 3\frac{|V|}{4} + \ldots) = O(|V|)$.

Quiz 1-4: [14 points] Parity Problems

Your friend has discovered a black magic algorithm to solve the all-pairs shortest paths problem on a weighted undirected graph $G = (V, E)$ in $O(V^2)$ time. However, due to some technical details, it only works on graphs with odd edge weights. You are tasked with designing a way to make the algorithm work on more general graphs. You may assume that all edge weights are natural numbers.

(a) **[5 points]** Describe an algorithm to produce vertex weights $h : V \to \{0, 1\}$ with the property that $h(u) - h(v) + w(u, v)$ is odd for every edge $(u, v) \in E$, where $w(u, v)$ is the weight of the edge (u, v). You may assume that there exists at least one function h satisfying the above property.

Solution. For each connected component, arbitrarily pick a vertex v and set $h(v) = 0$. Do any linear time traversal on that connected component starting from v, and for each traversed edge (u, w) set $h(w) = h(u)$ if $W(u, w)$ is odd, and $h(w) = 1 - h(u)$ otherwise.

(b) **[6 points]** Prove that a such a weighting $h : V \to \{0, 1\}$ exists if G contains no cycles with an odd number of even-weight edges.

Solution. Proof by induction on the number of edges.

The base case, $E = 0$, is trivially true.

Inductive step: Assume the hypothesis holds for a directed graph with k edges. Consider a graph $G = (V, E)$ with $k = |E| + 1$. If G has no even weight edges, then $h(v) = 0$ for all v is a valid assignment. Otherwise, pick an even weight edge, $e = (u, v)$ and delete it to obtain $G' = G \setminus e$. By the inductive hypothesis, we know G' has a valid assignment $h'(v)$.

If $h'(u) \neq h'(v)$ then $h = h'$ is a valid assignment on G as well, and we are done. Otherwise, we claim it must be the case that u and v are in separate connected components in G'. To see this, suppose for the sake of contradiction there is a path, P, from u to v in G'. Such a path cannot have an even number of even weight edges, for if it did, the cycle $P \cup e$ in G would have an odd number of even weight edges, which we have assumed not to be the case. Thus, P has an odd number of even weight edges which together force the constraint $h(u) \neq h(v)$ a contradiction.

In this case, we will take $h(v) = h'(v)$ for all v not in the connected component of u in G' and $h(v) = 1 - h'(v)$ otherwise. Now $h(u) \neq h(v)$, and so the constraint generated by e is satisfied, and it is easy to check that no other constraints become violated by flipping all values in a single connected component.

1 Thus, we have generated a valid assignment $h(v)$ for all cases and have completed the induction.

(c) **[3 points]** Design an algorithm to solve the all-pairs shortest paths problem on an undirected graph that does not contain any cycles with an odd number of even-weight edges in $O(V^2)$ time. You may assume that your friend's algorithm works as described.

Solution.

i. Reweight the edges by propagation in $O(E) = O(V^2)$ time.

ii. Run the magic algorithm as a black box in $O(V^2)$ time.

iii. For each path from step 2, the shortest path in the original graph is given by $W + h(v) - h(w)$.

Quiz 1-5: [15 points] K-Sum Subset

Suppose we have an array A with n elements, each of which is an integer in the range of $[0...50n]$. Give an $O(n \log n)$ algorithm that when given an integer $0 \leq t \leq 5 \cdot 50n$, determines whether there exist at most 5 (not necessarily distinct) elements of A which sum to t. That is, output "YES" if there exist $i_1, \ldots, i_k$ with $k \leq 5$ such that $A[i_1] + \ldots + A[i_k] = t$ and output "NO" otherwise.

Solution. Construct a polynomial $p(x)$ where the exponents of the polynomial correspond to elements of Array A.

$$P(x) = x^{A[0]} + x^{A[1]} + ... + x^{A[n-1]}$$

Idea: When we raise $P(x)$ to some power, we get a resuling polynomial that has a sum of terms, each of which is some variation of the form $x^{A[i]} * x^{A[j]+\cdots} = x^{A[i]+A[j]+\cdots}$ – where the number of terms in the sum is equal to the power we raised the polynomial to. For example, for $P^2(x)$, each term in the polynomial appears as $x^{A[i]+A[j]}$, and for $P^3(x)$, each term appears as $x^{A[i]+A[j]+A[k]}$. Therefore, we just need to check whether x^t appears in any power z of $P(x)$ for $0 \leq z \leq 5$, to determine whether there exist at most 5 elements of A that sum to t.

Algorithm:
while z $\leq$ 5:

1. Construct $p(x) = x^{A[0]} + x^{A[1]} + ... + x^{A[n-1]}$.
2. Compute $p^z(x)$ via FFT by repeated multiplications of p(x). (You can do this step by multiplying the polynomial from the previous iteration with P(x) again.
3. Scan through the resulting polynomial and check whether x^s is a term.

Analysis:
Constructing the polynomial $p(x)$ takes $O(n)$ time, as each element of A is in the range [0 ... 50n], so $p(x)$ is a degree of at most 50n. Each polynomial multiplication using FFT takes time $O(nlogn)$, and since we're only doing 5 multiplications max, the overall cost of this step is still $O(nlogn)$. Finally, scanning through $p^z(x)$ takes $O(n)$, because the degree for each scan is bounded by a constant (250n max for $p^5(x)$). Therefore, combining all of this together, the overall run time of this algorithm is $O(nlogn)$.