1 Amplifying the Success Probability of Randomized Algorithms

1.1 Item a

We define the algorithm $A'$ as follows: On input a graph $G$ the algorithm runs $A(G)$ for $t = 180 \log n$ times independently with fresh randomness and outputs the majority decision of these repetitions. The algorithm runs in time $O(T(n) \cdot \log n)$.

Let $X_1, \ldots, X_t$ be the random variables describing each independent execution where $X_i = 1$ if the algorithm answered correctly in execution $i$ (note that we are doing both the YES and NO cases simultaneously so we refer to success rather than the actual output of the algorithm). Notice that, by assumption, $E[X_i] = 2/3$ for every $i \in [t]$.

Define $X := \sum_{i=1}^{t} X_i$. By linearity of expectation, we have that $E[X] = \sum_{i=1}^{t} E[X_i] = 2t/3$. Thus, by the Chernoff bound, for all $a > 0$:

$$\Pr [ X \leq 2t/3 - a ] \leq \exp \left\{ \frac{-2a^2}{t} \right\}.$$ 

Setting $a = t/6$, we get

$$\Pr [ X \leq t/2 ] \leq \exp \left\{ \frac{-2(t/6)^2}{t} \right\} = \exp \left\{ -t/18 \right\} = \exp \left\{ -10 \log n \right\} = 1/n^{10}.$$ 

Whenever $X > t/2$ the algorithm answers correctly since the majority of the executions give the correct answer. Therefore the probability that the algorithm is correct is at least $1 - 1/n^{10}$ as required.

1.2 Item b

We describe the algorithm $A''$ on input a graph $G = ([n], E)$ and $k = O(1)$:

1. If $n = \Theta(k)$ then brute-force test for a $k$-clique: go over all possible choices of $k$ nodes and test whether they are a clique. If one is found then output “TRUE” and exit and otherwise output “FALSE” and exit.

2. If the algorithm has not exited yet, then partition the $n$ nodes of the graph into $k + 1$ arbitrary non-intersecting subsets $V_1, \ldots, V_{k+1}$.

3. For $i = 1$ to $k + 1$ do:
   (a) Let $V := \bigcup_{i \in [k+1] \setminus \{i\}} V_i$ and $G' := (V, E')$ where $E' := \{(u, v) \in E \mid u, v \in V\}$.
   (b) Let $b \leftarrow A'(G')$ (where $A'$ is the algorithm from the previous question).
   (c) If $b = “TRUE”,$ return $A''(G')$.

4. If the algorithm has not exited yet, return “FALSE”.
Analysis. We begin by analysing the running time of \( \mathcal{A}'' \). Notice that \( G' \) is a graph on \( k/(k+1) \cdot n \) nodes. Let \( t(n) \) be the running time of the algorithm given a graph with \( n \) nodes. We observe that (when \( n \) is not constant)
\[
t(n) \leq O \left( k \cdot T(n) \log n + t \left( \frac{k}{k+1} \cdot n \right) \right).
\]
Solving this recursion when \( k = O(1) \), and where the base layer takes time \( O(1) \) we get that \( t(n) = O(T(n) \log^2 n) \) as required.

We now analyse the correctness of the algorithm. First, note that the algorithm only ever returns “TRUE” at the base level of the recursion when it has found a \( k \)-clique. Since, by construction, this clique is a subset of the original graph if the algorithm, indeed, returns “TRUE” then the graph must contain a \( k \)-clique.

We now show that the algorithm errs with probability at most \( 1/n^9 \). Since we have already explained why the algorithm cannot err when there is no \( k \)-clique in the graph, we need only analyse the YES case. Any \( k \)-clique in the graph must be in at least one of the choices of \( V \) since it contains \( k \) nodes. If \( \mathcal{A}' \) fails on this choice of \( V \) then the algorithm will err. \( \mathcal{A}' \) errs with probability \( 1/n^{10} \).

This is true separately for every recursion level. By the union-bound, the probability that one of the critical executions of \( \mathcal{A}' \) fails is at most \( m/n^{10} \) where \( m \) is the recursion depth. Since we are cutting the number of nodes by a multiplicative constant \((k/(k+1)) \) in every round, the recursion depth is \( O(\log n) \). Therefore \( \mathcal{A}'' \) errs with probability at most \( O(\log n)/n^{10} < 1/n^9 \).

2 Convolution 3SUM to 3SUM

We recall that if 3SUM can be solved in time \( T(n, U) \), then colourful 3SUM can be solved in time \( O(n) + T(O(n), O(U)) \). We will show that convolution 3SUM with sets of size \( n \) and universe \( U \) can be reduced in linear time to colourful 3SUM with sets of size \( n \) and universe \( O(nU) \). Together with the previous statement, this shows that if 3SUM can be solved in time \( T(n, U) \) then convolution 3SUM can be solved in time \( O(n) + T(O(n), O(nU)) \).

Construction. Given a convolution 3SUM instance \( A, B, C \subseteq [-U, U] \) of size \( n \) we output the following colourful 3SUM instance \( A', B', C' \subseteq [-U', U'] \) of size \( n \) where \( U' = \Theta(nU) \).
\[
A' := \{ 4Ui + A[i] \mid i \in [n] \} \\
B' := \{ 4Ui + B[i] \mid i \in [n] \} \\
C' := \{ -4Ui + C[i] \mid i \in [n] \}
\]

Time and range analysis. Notice first that, indeed, \( A', B', C' \subseteq [-O(nU), O(nU)] \) since the indices are at most \( n \). Further notice that this transformation can be done in linear time given a single scan of each of the sets \( A, B, C \).

Correctness. Suppose that there exist \( i, j \in [n] \) such that \( A[i] + B[j] + C[i + j] = 0 \). Then:
\[
(4Ui + A[i]) + (4Uj + B[j]) + (-4U(i + j) + C[i + j]) \\
= 4U(i + j - (i + j)) + (A[i] + B[j] + C[i + j]) \\
= 0.
\]
Thus, the elements in $A', B', C'$ matching $A[i], B[j], C[i+j]$ sum to zero.

On the other hand, suppose that there exist $a \in A', b \in B', c \in C'$ with $a + b + c = 0$. Due to the way $a, b, c$ were constructed, there exist $i, j, k \in [n]$ such that $a = 4Ui + A[i], b = 4Uj + B[j]$ and $c = -4Uk + C[k]$. Therefore


If $i + j - k \neq 0$, then $|4U(i+j-k)| \geq 4U > 3U$. Since $A[i] + B[j] + C[k] \in [-3U, 3U]$, this sum cannot zero out the expression. In other words, in order for the above expression to be equal to 0 it must be that $i + j = k$. Once we zero out the expression $4U(i + j - k)$ we are left with the requirement that $A[i] + B[j] + C[k] = A[i] + B[j] + C[i + j] = 0$, completing the proof of correctness.

3 Triangle Detection to 3SUM

Recall that if 3SUM can be solved in time $T(n, U)$, then colourful 3SUM can be solved in time $O(n) + T(O(n), O(U))$. We will show that convolution 3SUM with sets of size $n$ and universe $U$ can be reduced in linear time to colourful 3SUM with sets of size $n$ and universe $O(nU)$. Therefore, we work with colourful 3SUM rather than vanilla 3SUM.

Construction. We transform a graph $G = ([n], E)$ with $|E| = m$ into a colourful 3SUM instance as follows:

$$A := \{ 6n^2 \cdot i + 2n \cdot j \mid (i, j) \in E \land i > j \}$$

$$B := \{ -6n \cdot i + j \mid (i, j) \in E \land i > j \}$$

$$C := \{ -6n^2 \cdot i - j \mid (i, j) \in E \land i > j \}$$

Time and range analysis. Since $i, j \in [n]$, we have that $A, B, C \subseteq [-O(n^3), O(n^3)]$ and since an element is added to one of the sets if and only if there exists an edge that corresponds to it, we have that $|A|, |B|, |C| = O(|E|) = O(m)$. Notice that this transformation can be done in linear time. Therefore, once we show correctness (i.e., that the colourful 3SUM instance is a YES instance if and only if there exists a triangle in the graph) we will have shown, as required, that if 3SUM can be solved in time $T(n, U)$ then triangle detection can be done in time $O(n) + T(O(m), O(n^3))$.

Correctness. We begin by showing that if the graph has a triangle, then the constructed 3SUM instance has a solution. Let $(i, j), (j, k), (i, k) \in E$ be a triangle in the graph with $i > j > k$ and let $a, b, c$ be the matching values in the sets respectively. Then

$$a + b + c = (6n^2 \cdot i + 2n \cdot j) + (-6n \cdot j + k) + (-6n^2 \cdot i - k) = 0$$

\(^\text{1}\) Amir: If we start from a colored triangle instance, we can simplify this construction (and analysis) a bit by removing the $i < j$ conditions.
We now show that if the constructed 3SUM instance has a solution then the graph has a triangle. Suppose that there exist \( a \in A, b \in B, c \in C \) with \( a + b + c = 0 \). Then by construction there exist edges \((a_1, a_2), (b_1, b_2), (c_1, c_2) \in E\) where \( a = 6n^2 \cdot a_1 + 2n \cdot a_2 \) and \( a_1 > a_2 \). The values \( b \) and \( c \) are similarly defined. We now show that \( a_1 = c_1, a_2 = b_1 \) and \( b_2 = c_2 \). Notice that

\[
0 = a + b + c \\
= 6n^2 \cdot a_1 + 2n \cdot a_2 - 2n \cdot b_1 + b_2 - 2n^2 \cdot c_1 - c_2 \\
= 6n^2 (a_1 - c_1) + 2n (a_2 - b_1) + (b_2 - c_2) \\
\]

Since \( a_1, a_2, b_1, b_2, c_1, c_2 \in [n] \) we have \( 2n(a_2 - b_1) + (b_2 - c_2) \in [-2n^2 - n, 2n^2 + n] \). It must, therefore, be that \( a_1 - c_1 = 0 \) since otherwise \( |6n^2 (a_1 - c_1)| > 2n^2 + n \) and so there is no way for the values to zero out the expression. Similarly, \( a_2 - b_1 = 0 \) since \( b_2 - c_2 \in [-n, n] \) which is not enough to zero out \( 2n(a_2 - b_1) \) when \( |a_2 - b_1| \geq 1 \). Finally, this reduces to the requirement that \( b_2 - c_2 = 0 \). Therefore \( a_1 = c_1, a_2 = b_1 \) and \( b_2 = c_2 \).

Since \((a_1, a_2), (b_1, b_2), (c_1, c_2) \in E\) and \( a_1 = c_1, a_2 = b_1 \) and \( b_2 = c_2 \) we have that \((a_1, a_2), (a_2, b_2), (a_1, b_2) \in E\). We show that this must be a triangle:

1. **Self-edges:** There cannot be a self edge in the triple, since, by construction and by the equalities we have proved: \( a_1 > a_2, a_2 = b_1 > b_2 \) and \( a_1 = c_1 > c_2 = b_2 \).

2. **Duplicates:**
   
   (a) \((a_1, a_2) \neq (a_2, b_2)\) and \((a_2, b_2) \neq (a_1, b_2)\) since \( a_1 > a_2 \).
   
   (b) \((a_1, a_2) \neq (a_1, b_2)\) since this would imply that \( a_2 = b_2 \), but we know that \( a_2 > b_2 \).

4 **3SUM to Sumset Size**

We describe an algorithm that solves 3SUM in given the ability to make 2 oracle calls to a sumset size oracle. For convenience we use the case of 3SUM’ with repetitions where we are given a set \( S \subseteq [-U, U] \) and need to find \( a, b, c \in S \) with \( a + b = c \).

On input a 3SUM’ with repetitions instance \( S \):

1. If \( 0 \in S \) output “YES”.
2. Let \( k := \text{SumsetSize} (S + S) \).
3. Let \( \ell := \text{SumsetSize} (S + (S \cup \{0\})) \).
4. Output “YES” if \( \ell < k + |S| \). Otherwise output “NO”.

**Analysis.** Clearly this algorithm runs in linear time if oracle calls are unit cost. Therefore we only need to show correctness.

Since we allow repetitions, and, surprisingly, \( 0 + 0 = 0 \), if \( 0 \in S \) then \( S \) is a “YES” instance, and the algorithm indeed outputs “YES”. We therefore henceforth assume that \( 0 \notin S \). Notice that in this case \((S + (S \cup \{0\}) \equiv (S + S) \cup S \)

- **"YES" Case:** Let \( a, b, c \in S \) be a 3SUM’ with repetitions solution. Then \( a + b \in (S + S) \) and \( a + b = c \in S \). Therefore \((S + S) \cap S \neq \emptyset \), implying that \( \ell = \left| S + (S \cup \{0\}) \right| < \left| S + S \right| + |S| = k + |S| \), and the algorithm will output “YES”.

• "NO" Case: Suppose that for every $a, b, c \in S$: $a + b \neq c$. Then, for every pair $a, b \in S$: $a + b \in (S + S)$ and $a + b \notin S$. Similarly, every $c \in S$ is not in $S + S$. Therefore $(S + S) \cap S = \emptyset$, implying that $\ell = |S + (S \cup \{0\})| = |(S + S) \cup S| = |S + S| + |S| = k + |S|$, and the algorithm will output “NO".