CS60-454
Design and Analysis of Algorithms
Winter 2017

Assignment 3

Due Date: March 23 (before lecture)

1. Let $w_1, w_2, \ldots, w_n$ be a sequence of non-negative numbers and M be a positive number such that $w_i \le M, 1 \le i \le n$.

   A *subdivision* of the sequence is a sequence of indices $i_1, i_2, \ldots, i_k$ such that:

   (a) $1 \le i_1 < i_2 < \cdots < i_k < n$,

   (b) $\sum_{s=1}^{i_1} w_s \le M, \sum_{s=i_j+1}^{i_{j+1}} w_s \le M, 1 \le j < k$, and $\sum_{s=i_k+1}^{n} w_s \le M$.

   An *optimal subdivision* is one that has the smallest k.

   Present a greedy algorithm which, when presented with $w_1, w_2, \ldots, w_n$ and M as input, produces an optimal subdivision for the sequence. Your algorithm should run in $O(n)$ time.

   Does the greedy algorithm work if the w_i's are allowed to be negative?

Solution:

A subdivision partitions the sequence into k subsequences.

To minimize k, we shall make each subdivision as long as possible.

The greedy strategy is thus:

starting from w_1, form the *longest* subsequence $w_1, w_2, \ldots, w_{i_1}$ such that $\sum_{s=1}^{i_1} w_s \le M$.

Then, starting from w_{i_1}, form the *longest* subsequence $w_{i_1+1}, w_{i_1+2}, \ldots, w_{i_2}$ such that $\sum_{s=i_1+1}^{i_2} w_s \le M$, and so on.

Since if $\sum_{s=1}^{n} w_s \le M$, the optimal subdivision is the null sequence and $k = 0$, we shall consider the cases where $k \ge 1$.

Theorem 1: Let ℓ be such that $\sum_{s=1}^{\ell} w_s \le M$ and $\sum_{s=1}^{\ell+1} w_s > M$. There exists an optimal subdivision $i_1, i_2, \ldots, i_k$ such that $i_1 = \ell$.

Proof:

Let $i_1, i_2, \ldots, i_k$ be an optimal subdivision. Then $\sum_{s=1}^{i_1} w_s \le M$

Since $\sum_{s=1}^{\ell+1} w_s > M$, therefore $i_1 \le \ell \le i_j$, where $2 \le j \le k$.

If $i_1 = \ell$, then the proof is complete.

Suppose $i_1 < \ell$.

If $i_j \le \ell$, for some $j \ge 2$,

then $\sum_{s=1}^{i_j} w_s \le \sum_{s=1}^{\ell} w_s \quad (\because i_j \le \ell)$

$\Rightarrow \sum_{s=1}^{i_j} w_s \le M \quad (\because \sum_{s=1}^{\ell} w_s \le M)$

$\Rightarrow i_j, i_{j+1}, \ldots, i_k$ is a subdivision that has at most $k - j + 1 \leq k - 1$ ($\because j \geq 2$) indices

$\Rightarrow i_1, i_2, \ldots, i_k$ is not an optimal subdivision, a contradiction!

Hence, $\ell < i_j, \forall j \geq 2 \Rightarrow i_1 < \ell < i_2$.

Since $\sum_{s=1}^{\ell} w_s \leq M$, $\sum_{s=\ell+1}^{i_2} w_s \leq \sum_{s=i_1+1}^{i_2} w_s \leq M$, $\sum_{s=i_j+1}^{i_{j+1}} w_s \leq M, 2 \leq j < k$, and $\sum_{s=i_k+1}^{n} w_s \leq M$,

$\ell, i_2, i_3, \ldots, i_k$ is also an optimal subdivision of the given sequence. $\square$

Theorem 2: Let $i_1, i_2, \ldots, i_k$ be an optimal subdivision of the sequence $w_1, w_2, \ldots, w_n$. Then, $i_2, \ldots, i_k$ is an optimal subdivision of the subsequence $w_{i_1+1}, w_{i_1+2}, \ldots, w_n$.

Proof:

Suppose to the contrary that $i_2, \ldots, i_k$ is not an optimal subdivision of the subsequence $w_{i_1+1}, w_{i_1+2}, \ldots, w_n$.

Let $j_1, j_2, \ldots, j_q$ be an optimal subdivision of $w_{i_1+1}, w_{i_1+2}, \ldots, w_n$, where $q < k - 1$.

Then $i_1, j_1, j_2, \ldots, j_q$ is a subdivision of $w_1, w_2, \ldots, w_n$.

As $q < k - 1 \Rightarrow q + 1 < k$, $i_1, i_2, \ldots, i_k$ is not an optimal subdivision of $w_1, w_2, \ldots, w_n$ which contradicts the assumption. $\square$

The greedy algorithm is as follows:

Algorithm `Subdivision`;

Input: A sequence of non-negative numbers $w_1, w_2, \ldots, w_n$ and a positive number M;

Output: An optimal subdivision, $i_1, i_2, \ldots, i_k$, of $w_1, w_2, \ldots, w_n$;

1. $k := 0$; $j := 0$; $sum := 0$;
2. **while** $(j < n)$ **do**

    repeat $j := j + 1$; $sum := sum + w_j$ **until** $(sum > M) \vee (j = n)$;

    if $(sum > M)$ **then**

        $k := k + 1$;

        $i_k := j - 1$;

        $j := j - 1$; $sum := 0$; // re-initialize sum

   endwhile;

Theorem 3: ***Algorithm*** `Subdivision` *correctly determines an optimal subdivision.*

Proof: By simple induction on k and Theorems 1 and 2 (I let you fill in the detail).

Theorem 4: ***Algorithm*** `Subdivision` *takes $O(n)$ time.*

Proof: Step 1 takes $O(1)$ time.

In Step 2, the **repeat** loop scans the list $w_1, w_2, \ldots, w_n$ once, spending $O(1)$ time on each $w_i, 1 \leq i \leq n$. The **repeat** loop thus takes $O(n)$ time.

The body of the **while** loop excluding the **repeat** loop takes $O(1)$ time. Since the **while** loop iterates k times, the body of the **while** loop excluding the **repeat** loop takes $O(k)$ time.

Algorithm `Subdivision` thus takes $O(1) + O(n) + O(k) = O(n + k) = O(n)$ time. $\square$

No, the greedy algorithm does not work if the w_i's are allowed to be negative. The following is a counterexample.

Let $(w_1, w_2, w_3) = (1, 1, -1)$ and $M = 1$.

The greedy algorithm produces the subdivision 1 (i.e. $k = 1$ and $i_1 = 1$) which subdivides the given sequence into $(1), (1, -1)$.

Since $1 + 1 + (-1) = 1 \leq M$, the optimal solution is the *null* sequence (i.e. $k = 0$) which gives no subdivision to the given sequence. $\blacksquare$

2. Let $H = \{h_j \mid 1 \leq j \leq n\}$ and S be two sets of real numbers such that $|S| = m$, where $n \leq m$. For each h_j in H, let s_j be a *distinct* number in S that is matched with h_j.

   (a) Prove that if $s_i < s_j$ and $h_i < h_j$, then $|h_i - s_i| + |h_j - s_j| \leq |h_i - s_j| + |h_j - s_i|$.

   (b) Present an $O(m \lg m + mn)$-time algorithm that matches each number h_j in H with a distinct number s_j in S such that the sum $\sum_{j=1}^{n} |h_j - s_j|$ is minimum.

Solution:

(a) We shall consider the case $s_i < h_i$ (the case $h_i \leq s_i$ is similar).

Since $s_i < s_j$ and $h_i < h_j$, three cases are to be considered:

(i) $s_i < s_j < h_i < h_j$: $|h_i - s_i| + |h_j - s_j|$

$$
\begin{aligned}
&= (h_i - s_i) + (h_j - s_j) \quad \text{(Definition of } |\;|) \\
&= (h_i - s_i) + (h_j - s_j) + ((s_i - s_j) - (s_i - s_j)) \\
&= ((h_i - s_i) + (s_i - s_j)) + ((h_j - s_j) - (s_i - s_j)) \\
&= (h_i - s_j) + (h_j - s_i) \\
&= |h_i - s_j| + |h_j - s_i| \quad \text{(Definition of } |\;|)
\end{aligned}
$$

(ii) $s_i < h_i < s_j < h_j$: $|h_i - s_i| + |h_j - s_j|$

$$
\begin{aligned}
&= (h_i - s_i) + (h_j - s_j) \quad \text{(Definition of } |\;|) \\
&= (h_i - s_i) + (h_j - s_j) + ((s_i - s_j) - (s_i - s_j)) \\
&= ((h_i - s_i) + (s_i - s_j)) + ((h_j - s_j) - (s_i - s_j)) \\
&= (h_i - s_j) + (h_j - s_i) \\
&= (h_i - s_j) + (h_j - s_i) + 2(s_j - h_i) - 2(s_j - h_i) \\
&= (2(s_j - h_i) - (s_j - h_i)) + (h_j - s_i) - 2(s_j - h_i) \\
&= (s_j - h_i) + (h_j - s_i) - 2(s_j - h_i) \\
&\leq (s_j - h_i) + (h_j - s_i) \quad (\because 2(s_j - h_i) > 0) \\
&= |h_i - s_j| + |h_j - s_i| \quad \text{(Definition of } |\;|)
\end{aligned}
$$

(iii) $s_i < h_i < h_j < s_j$: $|h_i - s_i| + |h_j - s_j|$

$$
\begin{aligned}
&= (h_i - s_i) + (s_j - h_j) \quad \text{(Definition of } |\;|) \\
&= (h_i - s_i) + (s_j - h_j) + (2(h_i - h_j) - 2(h_i - h_j)) \\
&= ((h_i - s_i) - (h_i - h_j)) + ((s_j - h_j) - (h_i - h_j)) + 2(h_i - h_j) \\
&= (h_j - s_i) + (s_j - h_i) + 2(h_i - h_j) \\
&= (s_j - h_i) + (h_j - s_i) - 2(h_j - h_i) \\
&\leq (s_j - h_i) + (h_j - s_i) \quad (\because 2(h_j - h_i) \geq 0)) \\
&= |h_i - s_j| + |h_j - s_i| \quad \text{(Definition of } |\;|)
\end{aligned}
$$

We thus have $|h_i - s_i| + |h_j - s_j| \le |h_i - s_j| + |h_j - s_i|$. $\square$

(b) First, based on Part (a), we sort both lists H and S into ascending order. Let the resulting lists be $h_i, 1 \le i \le n$, and $s_i, 1 \le i \le m$, respectively.

Let $A[k, \ell], 1 \le k \le n, 1 \le \ell \le m$, be the minimum cost (i.e. the minimum sum of absolute differences) for matching $h_i, 1 \le i \le k$, with $s_j, 1 \le j \le \ell$, so that each h_i has a matching s_j. For clarity, we shall call a matching that achieves the minimum cost an *optimal matching*.

Two cases are to be considered, namely matching s_ℓ with some h_i or not matching s_ℓ with any h_i:

- **Lemma 1:** In the matching with minimum cost $A[k, \ell]$, h_ℓ is matched with some s_k. Then
$$A[k, \ell] = A[k-1, \ell-1] + |s_k - h_\ell|.$$
Proof:
Let $(h_i, s_{j_i}), 1 \le i \le k$, be an optimal matching for matching $h_i, 1 \le i \le k$, with $s_j, 1 \le j \le \ell$ and s_ℓ is matched with some $s_i, 1 \le i \le k$.
If $s_{j_k} \ne s_\ell$ (i.e s_ℓ is not matched up with h_k), then it is matched up some $h_i, i < k$.
Since $i < k$ and $j_k < \ell$, by Part (a), $|h_i - s_{j_k}| + |h_k - s_\ell| \le |h_i - s_\ell| + |h_k - s_{j_k}|$.
Hence, s_ℓ can be matched up with h_k.
Then, $(h_i, s_{j_i}), 1 \le i \le k-1$, is a matching for matching $h_i, 1 \le i \le k-1$, with $s_j, 1 \le j \le \ell - 1$.
Suppose this matching is not optimal. Let its cost be C.
Let $(h_i, s_{\tilde{j}_i}), 1 \le i \le k-1$, be an optimal matching for matching $h_i, 1 \le i \le k-1$, with $s_j, 1 \le j \le \ell - 1$ and the optimal cost be $\tilde{C}$.
Then $\tilde{C} < C$
  $\Rightarrow \tilde{C} + |s_k - h_\ell| < C + |s_k - h_\ell|$
  $\Rightarrow (h_i, s_{j_i}), 1 \le i \le k$, is not an optimal match for matching $h_i, 1 \le i \le k$, with $s_j, 1 \le j \le \ell$, a contradiction!
Therefore, $(h_i, s_{j_i}), 1 \le i \le k-1$, is an optimal matching for matching $h_i, 1 \le i \le k-1$, with $s_j, 1 \le j \le \ell - 1$, i.e. $C = A[k-1, \ell-1]$.
Hence, $A[k, \ell] = A[k-1, \ell-1] + |s_k - h_\ell|$. $\square$

- **Lemma 2:** In the matching with minimum cost $A[k, \ell]$, h_ℓ is not matched with any s_i. Then
$$A[k, \ell] = A[k, \ell - 1].$$
Proof: Since h_ℓ is not matched with any s_i, the optimal matching matches $h_i, 1 \le i \le k$, with $s_j, 1 \le j \le \ell - 1$.
Hence, $A[k, \ell] = A[k, \ell - 1]$. $\square$

Since the above two cases are mutually exclusive, we have:

$$A[k, \ell] = \min\{A[k-1, \ell-1] + |s_k - h_\ell|, A[k, \ell-1]\}, 1 \le k \le \ell \le m.$$

For the base cases, since every h_i must be assigned a distinct s_j, we have:

$$A[k, \ell] = \infty, \text{ for } k > \ell \ge 0.$$

Moreover, if $k = 0$, then $A[k, \ell] = 0$.

We thus have the following recurrence:

$$A[k,\ell] = \begin{cases} 0, & k = 0; \\ \min\{A[k,\ell-1], A[k-1,\ell-1] + |s_k - h_\ell|\}, & 1 \le k \le \ell \le m; \\ \infty, & k > \ell \ge 0, \end{cases}$$

where $1 \le k \le n, 1 \le \ell \le m$.

Since the calculation of $A[k,\ell]$ depends on $A[k,\ell-1]$ and $A[k-1,\ell-1]$ which are to the left of $A[k,\ell]$, and above and to the left of $A[k,\ell]$, respectively, we can fill up the array $A[k,\ell], 1 \le k \le n, 1 \le \ell \le m$, in a *row-major order from left to right.*

The following is a pseudo-code of the algorithm:

Algorithm `Min-Absolute-Sum`;
Input: Two lists of real numbers $h_{[1..n]}$ and $s_{[1..m]}$;
Output: An optimal matching $(h_i, s_{j_i}), 1 \le i \le n$.
begin
Sort the list $h_{[1..n]}$ into ascending order;
Sort the list $s_{[1..m]}$ into ascending order;
for $\ell := 0$ **step** 1 **to** m **do** $A[0,\ell] := 0$;   // $A[k,\ell] = 0$, if $k = 0$
for $k := 1$ **step** 1 **to** n **do**
    for $\ell := k-1$ **step** -1 **to** 0 **do** $A[k,\ell] := \infty$;   // $A[k,\ell] = \infty$, if $k > \ell \ge 0$
for $k := 1$ **step** 1 **to** n **do**
    for $\ell := k$ **step** 1 **to** m **do**
        $A[k,\ell] := \min(A[k,\ell-1], A[k-1,\ell-1] + |s_k - h_\ell|)$;

/* Determine the optimal match */
$k := n$; $\ell := m$;
while $(k \ne 0)$ **do**
    if $(A[k,\ell] = A[k,\ell-1])$ **then**
        $\ell := \ell - 1$;   // continuing tracing from $A[k,\ell-1]$
    else output (h_k, s_ℓ);
        $k := k-1$; $\ell := \ell - 1$;   // continuing tracing from $A[k-1,\ell-1]$
endwhile;
end.

Theorem 3: ***Algorithm*** `Min-Absolute-Sum` *correctly generates an optimal match for the two lists of real numbers $h_{[1..n]}$ and $s_{[1..m]}$.*
Proof:
First, prove that the third **for** loop correctly computes $A[k,\ell], 1 \le k \le n, 1 \le \ell \le m$, by applying an induction on k and within the induction step applying an induction on ℓ.
Next, prove that an optimal matching $(h_k, s_{j_k}), 1 \le k \le n$, is correctly generated by applying an induction on k with $k = n$ as the base case.
As these induction proofs are simple, I let you fill in the detail. □

Theorem 4: ***Algorithm*** *`Min-Absolute-Sum` takes $O(m \lg m + mn)$ time.*

Proof:

Sorting takes $O(n \lg n) + O(m \lg m) = O(m \lg m)$ time because $n \le m$.

The first **for** loop takes $O(n)$ time.

The second **for** loop takes $\sum_{k=1}^{n} k = \frac{(n+1)n}{2} = O(n^2)$ time.

The body of the inner **for** loop of the third **for** loop takes $O(1)$ time per iteration, the inner **for** loop thus takes $O(m)$ time. Since the third **for** loop iterates n times, it takes $n \cdot O(m) = O(mn)$ time.

The **while** loop starts with $k = n$ and $\ell = m$ and iterates until $k = 0$.

Since during each iteration, at least one of k and ℓ is decrement by 1 and $k \le \ell$, the **while** loop iterates at most $m + n$ times. As the body takes $O(1)$ time, the **while** loop thus takes $O(m+n)$ time.

Hence **Algorithm** `Min-Absolute-Sum` takes $O(m \lg m) + O(n) + O(n^2) + O(mn) + O(m + n) = O(m \lg m + mn)$ time. [Note: $n \le m \Rightarrow n^2 \le mn$] $\square$

3. Let $G = (V, E)$ be a *simple* graph (a graph without self-loop and parallel edges) and $d_1, d_2, \ldots, d_n$ be the degrees of the vertices in G in *descending* order, where $n = |V|$. Let $v_i, 1 \le i \le n$, be such that d_i is the degree of v_i in G.

   (a) Prove that if $\exists j, 1 \le j \le d_1 + 1$ such that $\{v_1, v_j\} \notin E$, then $\exists \ell, 1 \le j < \ell \le n$ and $\exists u \in V - \{v_j, v_\ell\}$ such that $\{v_1, v_j\} \notin E \wedge \{u, v_\ell\} \notin E$ and $\{v_1, v_\ell\} \in E \wedge \{u, v_j\} \in E$.

   (b) Use Part (a) to prove that there exists a graph $G' = (V, E')$ such that $d_1, d_2, \ldots, d_n$ is also the degrees of the vertices in G' and $\{v_1, v_i\} \in E', 2 \le i \le d_1 + 1$.

   (c) Present an algorithm that on input $d_1, d_2, \ldots, d_n$ (not necessarily sorted) determines if there is a graph of which $d_1, d_2, \ldots, d_n$ are the degrees of its vertices. Your algorithm should run in $O(n \lg n + D)$ time, where $D = \sum_{i=1}^{n} d_i$.

Solution:

(a) Suppose $\exists j, 1 \le j \le d_1 + 1$ such that $\{v_1, v_j\} \notin E$.

Since $deg(v_1) = d_1$,

$$\begin{aligned}(v_1, v_j) \notin E &\Rightarrow \exists \ell, 1 \le d_1 + 1 < \ell \le n \text{ and } (v_1, v_\ell) \in E \\ &\Rightarrow \exists \ell, 1 \le j < \ell \le n \text{ and } (v_1, v_\ell) \in E. \quad (\because j \le d_1 + 1)\end{aligned}$$

Since $d_1, d_2, \ldots, d_n$ is in *descending* order,

$$\begin{aligned} j < \ell &\Rightarrow deg(v_j) \ge deg(v_\ell) \\ &\Rightarrow |\{w \in V - \{v_j, v_\ell\} \mid \{w, v_j\} \in E\}| \ge |\{w \in V - \{v_j, v_\ell\} \mid \{w, v_\ell\} \in E\}| \\ &\qquad \text{(definition of } degree\text{)} \\ &\Rightarrow |\{w \in V - \{v_j, v_\ell, v_1\} \mid \{w, v_j\} \in E\}| > |\{w \in V - \{v_j, v_\ell, v_1\} \mid \{w, v_\ell\} \in E\}| \\ &\qquad (\because \{v_1, v_j\} \notin E \text{ and } \{v_1, v_\ell\} \in E) \\ &\Rightarrow \exists u \in V - \{v_j, v_\ell, v_1\}, \{u, v_j\} \in E \text{ and } \{u, v_\ell\} \notin E \\ &\Rightarrow \exists u \in V - \{v_j, v_\ell\}, \{u, v_j\} \in E \text{ and } \{u, v_\ell\} \notin E. \end{aligned}$$

We thus have: $\exists \ell, 1 \le j < \ell \le n$ and $\exists u \in V - \{v_j, v_\ell\}$ such that $\{v_1, v_j\} \notin E \wedge \{u, v_\ell\} \notin E$ and $(v_1, v_\ell) \in E \wedge \{u, v_j\} \in E$.

(b) Let $A = \{v_i \mid 2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E\}$.

We shall apply induction on $|A|$.

(Induction basis) Let $|A| = 0$.

Then $|\{v_i \mid 2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E\}| = 0$

$\Rightarrow \{v_i \mid 2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E\} = \emptyset$ (Definition of $|\ |$)

$\Rightarrow \sim (\exists i)(2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E)$ (Definition of A)

$\Rightarrow (\forall i) \sim (2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E)$ (E16,E18,E15, 60-231 courseware)

$\Rightarrow (\forall i)(2 \le i \le d_1 + 1 \Rightarrow \{v_1, v_i\} \in E)$ (FE8, 60-231 courseware)

$\Rightarrow \{v_1, v_i\} \in E, 2 \le i \le d_1 + 1.$ (Equivalent notations)

Hence, graph G' is the graph G itself.

(Induction hypothesis) Suppose the assertion holds for $|A| = k - 1 (1 \le k \le d_1)$.

(Induction step) Let $|A| = k$.

$k \ge 1 \Rightarrow |A| \ge 1$

$\Rightarrow \{v_i \mid 2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E\} \ne \emptyset$ (Definition of $|\ |$)

$\Rightarrow \exists j, 1 \le j \le d_1 + 1, \{v_1, v_j\} \notin E$ (Definition of A)

$\Rightarrow \exists \ell, 1 \le j < \ell \le n, \exists u \in V - \{v_j, v_\ell\}$ such that

$\{v_1, v_j\} \notin E, \{u, v_\ell\} \notin E, \{v_1, v_\ell\} \in E$ and $\{u, v_j\} \in E.$ (by Part (a))

Consider removing edges $\{v_1, v_\ell\}, \{u, v_j\}$ from G, and adding edges $\{v_1, v_j\}, \{u, v_\ell\}$ to G.

Let the resulting graph be $G' = (V, E')$.

Then $deg_{G'}(v_1) = |\{w \in V \mid \{v_1, w\} \in E'\}|$ (Definition of *degree*)

$= |\{w \in V \mid \{v_1, w\} \in E\} - \{\{v_1, v_\ell\}\} \cup \{\{v_1, v_j\}\}|$

$= deg_G(v_1) - 1 + 1$

$= deg_G(v_1).$

Similarly, we can prove that $deg_{G'}(v_j) = deg_G(v_j), deg_{G'}(v_\ell) = deg_G(v_\ell)$ and $deg_{G'}(u) = deg_G(u)$.

Moreover, $\forall v \in V - \{v_1, v_j, v_\ell, u\}, \{w \in V \mid \{v, w\} \in E'\} = \{w \in V \mid \{v, w\} \in E\} \Rightarrow deg_{G'}(v) = deg_G(v)$.

We thus have: $d_1, d_2, \ldots, d_n$ is also the degree sequence of G'.

Let $A' = \{v_i \mid 2 \le i \le d_1 + 1 \wedge \{v_1, v_i\} \notin E'\}$.

Then $\{v_i \mid 2 \le i \le d_1+1 \wedge \{v_1, v_i\} \notin E'\} = \{v_i \mid 2 \le i \le d_1+1 \wedge \{v_1, v_i\} \notin E\} - \{\{v_1, v_j\}\}$

$(\because \{v_1, v_j\} \notin E$ and $\{v_1, v_j\} \in E')$

$\Rightarrow |A'| = |A| - 1$

$\Rightarrow |A'| = k - 1.$

By the induction hypothesis, $d_1, d_2, \ldots, d_n$ is also the degree sequence of a graph $G'' = (V, E'')$ such that $\{v_1, v_i\} \in E'', 2 \le i \le d_1 + 1$.

Hence, by the Principle of mathematical induction, the assertion follows. ■

(c) For ease of explanation, we shall give a definition first.

Definition: A sequence of non-negative integers is a *degree sequence* of an undirected simple graph if consists of the degrees of the graph.

Our algorithm is based on the following lemma.

Lemma 1: Let $d_1, d_2, \ldots, d_n$ be a sequence of non-negative integers in *descending* order. Then $d_1, d_2, \ldots, d_n$ is a degree sequence if and only if $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n$ is a degree sequence.

Proof:

$\Rightarrow$) Let $d_1, d_2, \ldots, d_n$ be a degree sequence of a graph $G = (V, E)$.
Since it is in descending order, by Part (b), there exists a graph $G' = (V, E')$ such that $d_1, d_2, \ldots, d_n$ is also a degree sequence of G' with $\{v_1, v_i\} \in E', 2 \leq i \leq d_1 + 1$.
Let $d_i = deg_{G'}(v_i), 1 \leq i \leq n$.
Consider removing vertex v_1 from G'. Let the resulting graph be $G'' = (V - \{v_1\}, E'')$.
Since removing v_1 from G' also removes the edges $\{\{v_1, v_i\} \mid 2 \leq i \leq d_1 + 1\}$ from G', $\forall i, 2 \leq i \leq d_1 + 1, deg_{G''}(v_i) = deg_{G'}(v_i) - 1 = d_i - 1$.
Since $\forall i, d_1 + 1 < i \leq n$, no edge incident on v_i is removed, $deg_{G''}(v) = deg_{G'}(v)$.
We thus have: $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n$ is a degree sequence of G''.

$\Leftarrow$) Let $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n$ be a degree sequence of a graph $G = (V, E)$.
Let $d_i - 1 = deg_G(v_i), 2 \leq i \leq d_1 + 1$, and $d_i = deg_G(v_i), d_1 + 2 \leq i \leq n$.
Consider adding a new vertex v_1 and edges $\{v_1, v_i\}, 2 \leq i \leq d_1 + 1$, to G. Let the resulting graph be $G' = (V \cup \{v_1\}, E')$.
Then $deg_{G'}(v_1) = (d_1 + 1) - 2 + 1 = d_1$,
$\forall i, 2 \leq i \leq d_1 + 1, deg_{G'}(v_i) = deg_G(v_i) + 1 = (d_i - 1) + 1 = d_i$, and
$\forall i, d_1 + 2 \leq i \leq n, deg_{G'}(v_i) = deg_G(v_i) = d_i$.
Hence, $d_1, d_2, \ldots, d_n$ is a degree sequence of G'. $\square$

Key idea:

First, we sort the input sequence into descending order. Let the resulting sequence be $d_1, d_2, \ldots, d_n$. Remove d_1 and replace $d_i, 2 \leq i \leq d_1 + 1$, with $d_i - 1$. This results in the sequence $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$. By Lemma 1, $d_1, d_2, \ldots, d_n$ is a degree sequence if and only if $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$ is a degree sequence.

Next, sort the sequence $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$ into descending order and repeat the above steps until either a sequence of 0's or a sequence $\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_k$ such that $\tilde{d}_1 > k - 1$ is obtained.

In the former case, the sequence of 0's must be derived from a sequence of the form $h, 1, 1, \ldots, 1 (h 1's)$. The latter sequence is the degree sequence of a complete bipartite graph $K_{1,h}$. By Lemma 1, the input sequence is also a degree sequence.

In the latter case, as no simple graph can have a vertex of degree larger than the number of other vertices in the graph, the sequence is not a degree sequence. By Lemma 1, the input sequence is also not a degree sequence.

Since repeatedly sorting sequences could take $O(n^2 \lg n)$ time which is undesirable, we shall use a compacted representation of sequence which is defined as follows.

Definition: Let $d_1, d_2, \ldots, d_n$ be a sequence of non-negative integers with $d_1 = \max\{d_i \mid 1 \leq i \leq n\}$.

A sequence $n_{d_1}, n_{d_1-1}, \ldots, n_1, n_0$ is a ***compact sequence*** of $d_1, d_2, \ldots, d_n$ if $n_j, d_1 \geq j \geq 0$, is the number of occurrences of integer j in $d_1, d_2, \ldots, d_n$.

Specifically, n_{d_1} = the number of occurrences of d_1 in $d_1, d_2, \ldots, d_n$,

n_{d_1-1} = the number of occurrences of $d_1 - 1$ in $d_1, d_2, \ldots, d_n$,

n_{d_1-2} = the number of occurrences of $d_1 - 2$ in $d_1, d_2, \ldots, d_n$,

$\vdots$

n_1 = the number of occurrences of 1 in $d_1, d_2, \ldots, d_n$,

n_0 = the number of occurrences of 0 in $d_1, d_2, \ldots, d_n$.

Notice that $n_{d_1} \geq 1$.

[e.g. For $9, 9, 9, 8, 8, 6, 6, 6, 6, 4, 4, 4, 3, 3, 3, 3, 3, 1, 1, 1, 1$, $d_1 = 9$, and

$n_9 = 3, n_8 = 2, n_7 = 0, n_6 = 4, n_5 = 0, n_4 = 3, n_3 = 5, n_2 = 0, n_1 = 4, n_0 = 0$.

The compact sequence is thus: $3, 2, 0, 4, 0, 3, 5, 0, 4, 0$.]

Starting with $\Delta = 9$,

to remove 9, we have to subtract a 1 from each of the following 9 integers, resulting in the sequence: $8, 8, 7, 7, 5, 5, 5, 5, 5, 3, 4, 4, 3, 3, 3, 3, 3, 1, 1, 1, 1$ which is

$8, 8, 7, 7, 5, 5, 5, 5, 5, 4, 4, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1$ in decreasing order. $\cdots$ (I)

The corresponding compact sequence is $2, 2, 0, 4, 2, 6, 0, 4, 0$ which can be calculated as follows:

Let $\Delta = d_1$. First, reduce n_Δ by 1 (e.g. $n_9 = 3 - 1 = 2$) because the first occurrence of $\Delta (= d_1)$ is removed.

Then find the index k such that $\sum_{\Delta \geq j > k} n_j < \Delta \leq \sum_{\Delta \geq j \geq k} n_j$.

- For $j, \Delta \geq j > k$, since each of the n_j occurrences of j is reduced by 1, they all become occurrences of $j - 1$. As a result, n_j becomes n_{j-1} for $\Delta \geq j > k$.
- For n_k, $(\Delta - \sum_{\Delta \geq j > k} n_j)$ of the n_k occurrences of k have been reduced by 1, they all become occurrences of $k - 1$ while $\sum_{\Delta \geq j \geq k} n_j - \Delta$ of the n_k occurrences of k remain unchanged. As a result, n_{k-1} is increased to $n_{k-1} + (\Delta - \sum_{\Delta \geq j > k} n_j)$ while n_k is reduced to $\sum_{\Delta \geq j \geq k} n_j - \Delta$. But n_{k+1} occurrences of $k + 1$ have been turned into occurrences of k, n_k thus becomes $n_{k+1} + (\sum_{j=1}^{k} n_j - \Delta)$.
- For $k - 2 \geq j \geq 0$, n_j remains unchanged.

[e.g. In the above example, $\Delta = 9$. After reducing n_9 by 1, the compact sequence becomes $2, 2, 0, 4, 0, 3, 5, 0, 4, 0$.

Since $\sum_{9 \geq j \geq 5} n_i = 2 + 2 + 0 + 4 + 0 = 8 < 9 < 11 = 2 + 2 + 0 + 4 + 0 + 3 = \sum_{9 \geq j \geq 4} n_j$, $k = 4$ and the $2 (= n_9)$ occurrences of 9 are turned into 2 occurrences of 8;

the $2 (= n_8)$ occurrences of 8 are turned into 2 occurrences of 7;

the $4 (= n_4)$ occurrences of 6 are turned into 4 occurrences of 5;

Since $\Delta - \sum_{9 \geq j \geq 5} n_j = 9 - 8 = 1$, *one* of the occurrences of 4 is turned into *one* occurrence of 3 while $2\ (= 11 - 9 = \sum_{9 \geq j \geq 4} n_j - 9)$ of the occurrences of 4 remain unchanged.

As a result, the number of occurrences of 4 becomes $2 (= 0 + 2 = n_5 + (\sum_{9 \geq j \geq 4} n_j - \Delta))$ and the number of occurrences of 3 becomes $6 (= 5 + 1 = n_3 + (\Delta - \sum_{9 \geq j \geq 5} n_j)$.

The number of occurrences of 2, 1 and 0 remain unchanged.

Hence, the n_j values for the updated sequence (I) are $n_8 = 2, n_7 = 2, n_6 = 0, n_5 = 4, n_4 = 2, n_3 = 6, n_2 = 0, n_1 = 4, n_0 = 0$. Notice that $n_9 = 0$ and is omitted. The corresponding compact sequence is thus $2, 2, 0, 4, 2, 6, 0, 4, 0$.]

The following is a pseudo-code of the algorithm.

Algorithm `DegreeSeq`;

Input: A sequence of non-negative integers $d_i, 1 \le i \le n$;

Output: $\begin{cases} Yes, & \text{if the input sequence is a degree sequence;} \\ No, & \text{otherwise.} \end{cases}$

```
begin
for i := 1 step 1 to n do read(d_i);
Sort the sequence d_i, 1 ≤ i ≤ n, into descending order;

/* determine the compact sequence n_{d_1+1}, ..., n_1, n_0 of d_1, d_2, ..., d_n */
Δ := d_1;    // d_1 is the largest degree
for j := 0 step 1 to d_1 + 1 do n_j := 0;    // initialize n_j, 0 ≤ j ≤ d_1 + 1
i := 1; j := Δ;
while (i ≤ n) do    // determine n_j, 1 ≤ j ≤ d_1
    if (d_i = Δ) then
        n_j := n_j + 1; i := i + 1;    // n_j = # of vertices of degree Δ(= j)
    else    // start the next n_j
        Δ := Δ − 1; j := j − 1;    // note: Δ = j
endwhile;

/* Apply Part (b) */
j := Δ := d_1;
while (Δ > 0) do
    while (n_j > 0) do    // the current largest degree Δ exists
        n_j := n_j − 1;    // remove the first occurrence of Δ;
        sum := 0; k := j + 1;
        repeat    // look for n_k such that Σ_{Δ≥i≥k+1} n_i < Δ ≤ Σ_{Δ≥i≥k} n_i
            if (k > 0) then k := k − 1; sum := sum + n_k
                    else write("No, the input sequence is not a degree sequence");
                        stop;
        until (sum ≥ Δ);

        /* (n_k − (sum − Δ)) occurrences of k have become k − 1 */
        n_{k−1} := n_{k−1} + (n_k − (sum − Δ));    // add them to the group n_{k−1}
        /* (sum − Δ) occurrences of k remain unchange */
        n_k := n_{k+1} + (sum − Δ);    // add the n_{k+1} group to it
        if (Δ > k) then
            /* all n_i, Δ ≥ i ≥ k + 2, occurrences of i have become i − 1 */
```

for $i := \Delta$ **step** -1 **downto** $k+1$ **do** $n_i := n_{i+1}$;

endwhile;

$j := \Delta := \Delta - 1$; // decrement both Δ and j by 1

endwhile; // stop when $\Delta = 0$

write("*Yes, the input sequence is a degree sequence*");

end.

Lemma 2: *The second **for** loop and the first **while** loop correctly construct the compact sequence $n_j, 1 \leq j \leq d_1 + 1$, of the input sequence $d_1, d_2, \ldots, d_n$.*

Proof: By a simple induction on j. I let you fill in the detail. $\square$

Lemma 3: *When execution of the **repeat** loop inside the second **while** loop terminates normally (i.e. it is not aborted at the **stop** statement), $sum = \sum_{\Delta > i \geq k} n_i$ and $\sum_{\Delta \geq i \geq k+1} n_i < \Delta \leq sum$.*

Proof: By a simple induction on k. I let you fill in the detail. $\square$

Lemma 4: Let $n_\Delta + 1, n_j, \Delta - 1 \geq j \geq 0$, be the compact sequence of a decreasing sequence $\tilde{d}_\ell, \tilde{d}_{\ell+1}, \ldots, \tilde{d}_n$, where $\Delta = \tilde{d}_\ell$. Then $\tilde{n}_j, \Delta \geq j \geq 0$, is the compact sequence of the sequence $\tilde{d}_{\ell+1} - 1, \tilde{d}_{\ell+2} - 1, \ldots, \tilde{d}_{\ell+\tilde{d}_\ell} - 1, \tilde{d}_{\ell+\tilde{d}_\ell+1}, \ldots, \tilde{d}_n$, such that

$$\begin{cases} \tilde{n}_i = n_{i+1}, \Delta \geq i \geq k+1, \quad \text{(note: } n_{\Delta+1} = 0) \\ \tilde{n}_k = n_{k+1} + (sum(k) - \Delta), \\ \tilde{n}_{k-1} = (\Delta - sum(k+1)) + n_{k-1}, \\ \tilde{n}_i = n_i, k-2 \geq i \geq 0, \end{cases}$$

where $sum(h) = \sum_{\Delta \geq j \geq h} n_j, 0 \leq h \leq \Delta$, $sum(k{+}1) < \Delta \leq sum(k)$, and $sum(\Delta{+}1) = 0$.

Proof:

The sequence $\tilde{d}_{\ell+1} - 1, \tilde{d}_{\ell+2} - 1, \ldots, \tilde{d}_{\ell+\tilde{d}_\ell} - 1, \tilde{d}_{\ell+\tilde{d}_\ell+1}, \ldots, \tilde{d}_n$ can be obtained from the sequence $\tilde{d}_\ell, \tilde{d}_{\ell+1}, \ldots, \tilde{d}_n$, where $\Delta = \tilde{d}_\ell$, by removing the first integer $\tilde{d}_\ell$ and then subtract 1 from each of the following $\tilde{d}_\ell$ integers $\tilde{d}_{\ell+i}, 1 \leq i \leq \tilde{d}_\ell$.

After removing d_ℓ, $n_j, \Delta \geq j \geq 0$, is the compact sequence of the resulting sequence $\tilde{d}_{\ell+1}, \tilde{d}_{\ell+2}, \ldots, \tilde{d}_n$.

Since $sum(k+1) < \Delta \leq sum(k)$, $\tilde{d}_{\ell+\tilde{d}_\ell} = k$. It follows that after subtracting 1 from each of the first $\tilde{d}_\ell$ integers $\tilde{d}_{\ell+i}, 1 \leq i \leq \tilde{d}_\ell$, in the resulting sequence,

- the n_i occurrence of i have all been reduced to $i-1, \Delta \geq i \geq k+1$;
- $(\Delta{-}sum(k{+}1))$ of the n_k occurrence of k have been reduced to $k{-}1$ while $(sum(k){-}\Delta)$ occurrence of k remain unchanged. The number of occurrences of k is then $n_{k+1} + (sum(k) - \Delta)$, and the number of occurrences of $k-1$ is then $(\Delta - sum(k+1)) + n_{k-1}$.
- $n_i, k-2 \geq i \geq 0$, remains unchanged.

Hence, $\tilde{n}_j, \Delta \geq j \geq 0$, such that:

$$\begin{cases} \tilde{n}_i = n_{i+1}, \Delta \geq i \geq k+1, \quad \text{(note: } n_{\Delta+1} = 0) \\ \tilde{n}_k = n_{k+1} + (sum(k) - \Delta), \\ \tilde{n}_{k-1} = (\Delta - sum(k+1)) + n_{k-1}, \\ \tilde{n}_i = n_i, k-2 \geq i \geq 0, \end{cases}$$

is the compact sequence of the sequence $\tilde{d}_{\ell+1} - 1, \tilde{d}_{\ell+2} - 1, \ldots, \tilde{d}_{\ell+\tilde{d}_\ell} - 1, \tilde{d}_{\ell+\tilde{d}_\ell+1}, \ldots, \tilde{d}_n$.

$\square$

Lemma 5: *At the end of each iteration of the inner **while** loop of the second **while** loop,* $d_1, d_2, \ldots, d_n$*, is a degree sequence if and only if* $n_j, d_1 \geq j \geq 0$*, with leading 0's omitted is the compact sequence of a degree sequence.*

Proof: By a simple induction on the *iteration number*, using Lemmas 1, 2 and 4. I let you fill in the detail. □

Theorem 6: *Algorithm* `DegreeSeq` *reports "Yes" if the input sequence is a degree sequence and reports "No" if the input sequence is not a degree sequence.*

Proof:

When execution of the second **while** loop terminates normally, $n_j = 0, d_1 \geq j \geq 1$, and $n_0 > 0$. This is the compact sequence (with leading 0's omitted) of the degree sequence of a simple graph consisting of n_k isolated vertices. By Lemma 4, the input sequence is a degree sequence. Hence the algorithm correctly report "*Yes*".

On the other hand, if execution of the second **while** loop is aborted, it happens in the **repeat** loop with the conditions: $k = 0$ and $sum < \Delta$.

These conditions implies that $\sum_{\Delta \geq j \geq 1} n_j < \Delta$ which implies that if $n_j, \Delta \geq j \geq 1$, is the compact sequence of a degree sequence of a simple graph, then the graph would have a vertex of degree Δ and have less than Δ vertices. Since such a simple graph does not exist, by Lemma 5, the input sequence is not a degree sequence. Hence the algorithm correctly report "*No*". □

Theorem 7: ***Algorithm*** `DegreeSeq` *takes* $O(n \lg n + D)$ *time, where* $D = \sum_{i=1}^{n} d_i$*.*

Proof:

The first **for** loop takes $O(n)$ time. Sorting the input list takes $O(n \lg n)$ time. The second **for** loop takes $O(d_1) = O(n)$ time. The first **while** loop takes $O(d_1 + n) = O(n)$ time.

The second **while** loop and its inner **while** loop together iterates n times, once for each of the d_i's in the input sequence.

For the iteration corresponding to d_i, the **repeat** loop takes $O(\Delta) = O(d_i)$ time as it involves at most Δ n_j's and $\Delta \leq d_i$. Similarly, the **for** loop takes $O(\Delta) = O(d_i)$ time. The remaining statements takes $O(1)$ time. Therefore, each iteration takes a total of $O(d_i)$ time for a distinct d_i in the input sequence. The second **while** loop and its inner **while** loop thus take $\sum_{i=1}^{n} O(d_i) = O(\sum_{i=1}^{n} d_i) = O(D)$ time.

Hence, **Algorithm** `DegreeSeq` takes $O(n) + O(n \lg n) + O(n) + O(n) + O(D) = O(n \lg n + D)$ time, where $D = \sum_{i=1}^{n} d_i$. ■