2.1 Understand

1) Input: $G=(V,E,w)$ a weighted connected undirected graph and ℓ the length of the shortest path in G from s to d where $s=v_1$ and $d=v_k$ in a sequences of vertices $v_1, v_2, \ldots, v_k$.

Output: "yes" if $(v_i, v_{i+1}) \in E$ for all $1 \leq i < k$ and $\sum_{i=1}^{k-1} w(v_i, v_{i+1}) = \ell$

"no" otherwise

2) Input: $G=(V,E,w)$ a weighted connected undirected graph and let ℓ be the length of the shortest cycle in G that visits every vertex.

Output: "yes" if V' is a permutation of V such that $(v'_i, v'_{i+1}) \in E$ for all $1 \leq i < n$, $(v'_n, v'_1) \in E$ and $w(v'_n, v'_1) + \sum_{i=1}^{n} w(v'_i, v'_{i+1}) = \ell$

3) We can prove that Dijkstra's algorithm and Prim's algorithm has the same exact structure, so it only uses two separate loops. The first loop iterates through as many vertices that are not in your tree which is $\Theta(|V|)$ and the second loop iterates for $|V|$ times and iterates through each vertex v that is connected to V which is $\Theta(|E|)$. Thus, we can prove that Dijkstra's algorithm runs in polynomial time.

4) TravellingSalesmanVerifier($G=(V,E,w)$, ℓ, W)
if W is not a permutation of V, say "no"
if $\exists_{v' \in W}$: $(v'_i, v'_{i+1}) \notin E$ for all $1 \leq i < n$, say "no"
if $(v'_n, v'_1) \notin E$ for $v' \in E$, say "no"
if $w(v'_n, v'_1) + \sum_{i=1}^{n} w(v'_i, v'_{i+1}) = \ell$, say "no"
else say "yes"

5) "yes" → "yes"
Assume there is a cycle of length ℓ that visits every vertex in G and is the shortest path. Assume W is a permutation of V in the order that is visited along the shortest cycle in G. Because W is a permutation of V, it passes the first test. Because W has every vertex visited in the shortest cycle, it passes the second, third, and final tests.

"no" → "no"
Assume there is no cycle of length ℓ that visits every vertex in G and is the shortest path. Assume W is a permutation of V that visits every vertex along a cycle in G. Because W is a permutation of V that visits every vertex in a cycle, the first, second, and third tests must be true. However, because there is no cycle in G of length ℓ and W is a permutation of V that visits every vertex in a cycle in G, we say "no" for the final test.

Proof:
1st test checks if W is a permutation of V which runs in $\Theta(|E|)$
2nd test checks all edges in E which still runs in $\Theta(|E|)$
3rd test looks for a specific edges which is $\Theta(1)$.
Final test is the sum of all weights n times which is $\Theta(n)$
We can conclude that this runs in polynomial time.

2.2 Explore

1) To maximize the sum of products of both sets A and B, A and B should be in either ascending or descending order simultaneously. Thus, $A = \{1, 10, 100, 1000\}$ and $B = \{8, 12, 16, 42\}$ or $A = \{1000, 100, 10, 1\}$ and $B = \{42, 16, 12, 8\}$

2) A and B should be both in ascending order or A and B should both be in descending order.

3) Assume A is a list in ascending order such that $A[1] < A[2] < \ldots < A[n]$.
Let S represent the set B such that $B[1] < B[2] < \ldots < B[n]$
Let O represent the optimal set for B in respect to A.

Case 1: $O_i < S_i$
This is a contradiction because O has a lower value than our set is already supposed to select the lowest element for B. This is an impossible case.

Case 2: $O_i = S_i$
In this case, both sets share the same value. We can alter O by swapping our value with the element selected in O, so that we would now have an optimal solution.

Case 3: $O_i > S_i$
In this case, O has selected an element larger than ours which would mean the optimal set is not in ascending order. If $B[j]$ is at the same index of $B[i]$ in the optimal set where $j > i$, then we would be put in a scenario where the element $B[j]$ in our optimal set is at the same spot as $B[i]$ is in our set, so we must swap the two elements to bring O to the optimal solution

Theorem: Because A is in ascending order and B always returns the lowest element first to maximize the sum $\sum_{i=1}^{n} A_i B_i$

Proof: (continued)

Assume that O selects our element o_i the j^{th} element where $j > i$. Then

$$c(O) - c(O') = \sum_{i=1}^{n}(n-i+1)o_i - \sum_{i=1}^{n}(n-i+1)o_i'$$

$$= (n-i+1)o_i + (n-j+1)s_i - (n-i+1)s_i - (n-j+1)o_i$$

$$= (n-i+1)(o_i - s_i) + (n-j+1)(s_i - o_i)$$

$$= (n-i+1)(o_i - s_i) - (n-j+1)(o_i - s_i)$$

$$= ((n-i+1) - (n-j+1))(o_i - s_i)$$

$$= (j-i)(o_i - s_i)$$

Because $j > i$ and $o_i > s_i$, then the difference in cost is positive which contradicts the assumption that O was optimal.

As case 1 and 3 are contradictions and case 2 is our own viable solution, it means our proposed set B where B is in ascending order when A is is optimal.

2.3 Expand

1) Input: A set of Pokemon cards S that consists of n different cards, a set of m number of packs P, and for $1 \leq i \leq m$ $P[i] \subseteq S$.
Output: "yes" if there exists a set $P' \subseteq P$ and a minimum number of packs k that covers S.

2a) PokemonVerifier($S[1..n]$, $P[1..m]$, k, W)
if $W \not\subseteq P$ say "no"
if $|W| > k$ say "no"
if $W[1] \cup W[2] \cup \ldots \cup W[k] \subset S$ say "no"
else say "yes"
EndVerifier

2b) "yes" → "yes"
Assume there is a minimum number of packs k that will cover S and assume W is the set that contains these packs. Because W contains a specific number of packs within $P[1..m]$, that is the minimum number of packs, then we pass the first two tests. Because the packs in W cover S, then W is not just a proper subset of S, so we pass the final test.

"no" → "no"
Assume there is no minimum number of packs k that will cover S and assume $W \subseteq P$ and is size k. This passes the first two tests as $W \subseteq P$ and $|W| = k$. However, this fails the final test as there is no k number of packs that will cover S, so we say no.

Polynomial Proof:
For the 1st test, testing for a subset is $O(n^2)$
For 2nd test, checking & comparing the size of W is $O(1)$
For final test, checking if all elements in W converge with S is still $O(n^2)$.

3a) Vertex Cover $\leq_p$ Pokemon

$m(G=(V,E), K)$

Let $S = E$

Let $P = V$

Let $k' = k$

return $m(S, P, k')$

3b) "yes" $\rightarrow$ "yes"

Assume that G has a set of vertices V' of size K such that $\forall_{(u,v)\in E}: u \in V'$ or $v \in V'$ and $V' \subseteq V$. Because P is the set of vertices such that $P = V$ and covers all of the edges and S contains all of the edges that is covered by E, then we can confirm there is a relationship.

"yes" $\leftarrow$ "yes"

Assume P has the set of vertices of V' such that $P = V$ that covers all of the edges with k paths and S contains all of the edges in E. Since G has a set of vertices that is the subset of V of size k that also covers all of the edges, then we can conclude that there is a relationship between these two problems.

2.4 Challenge

1) Input: Let $D[1...n]$ be a list of distances the zombies are away from the trio and let $S[1...n]$ be a list of speeds each zombie is moving on.
Output: Let $Z \subseteq \{1, ..., n\}$, a set of indexes or zombies in an order that achieves m, the maximum amount of time that lets me, Alice, and Bob survive the longest.

2) Counter Example:
Let $D = \{4, 6, 8\}$
Let $S = \{1, 3, 4\}$
Killing the closest zombie won't lead to a valid solution. $Z[1]$ will be shot first, then $Z[2]$ will be shot due to being closer. However, $Z[3]$ can reach the trio in 2 seconds. This doesn't lead to a valid solution as it takes 3 seconds to shoot 3 zombies, so the trio will be a second too late.

3) Counter Example:
Let $D = \{2, 100\}$
Let $S = \{2, 10\}$
Killing the fastest zombie won't lead to a valid solution. $Z[2]$ will be shot first as it is the fastest, but $Z[1]$ will be able to reach the group as it would take another second to shoot $Z[1]$ and $Z[1]$ can already reach the group in a second.

4) The zombies should be shot in an ascending order in respect to time. If T is a list of the time it takes for each zombie to reach the group as in the distance divided by speed for each zombie represented as d_i/s_i at each index is in ascending order. So, $T_1 \leq T_2 \leq \ldots \leq T_n$ will lead to the optimal solution.

5) Let S represent the set T as stated before such that $T[1] \leq T[2] \leq \ldots \leq T[n]$.
Let O represent the optimal set of zombies to kill in respect to time.
Let i represent the index where s_i and o_i disagree.
Case 1: $o_i < s_i$;
This is a contradiction because it contradicts that we always select the zombie who reaches us in the shortest time.

Case 2: $o_i = s_i$;
In this case, both sets share the same value, so there is no disagreement here.

Case 3: $o_i > s_i$;
In this case, O has selected a time larger than ours which would mean the optimal set is not in ascending order. Let j represent another index such that $j > i$. Because $T[j]$ in O is greater than $T[i]$ in our set, then a $T[j]$ does exist further in the optimal set such that $T[j] = T[i]$ in S. Thus, we must swap the two elements to bring O to the optimal solution.

Case 3 (continued):
This is also a contradiction. Because $s_i < o_i$ and s_i exists further into O at the jth index where $j > i$, it would take the group $j-i$ seconds in order to kill that zombie. That zombie already had i seconds to move, so its current time is $Z[j]-i$. As such, if $Z[j]-i < j-i$ or $Z[j] < j$, then this does not lead to a valid solution. Else, this leads to a valid, but not optimal solution. As such, a swap is necessary to bring O to an optimal set.

Theorem: The group always shoots the zombie who takes the least amount of time to reach the group.
Proof: (continued)
Assume O selects our zombie time as the jth time where $j > i$. Then,

$$c(O) - c(O') = \sum_{i=1}^{n}(n-i+1)o_i - \sum_{i=1}^{n}(n-i+1)o_i'$$
$$= (n-i+1)o_i + (n-j+1)s_i - (n-i+1)s_i - (n-j+1)o_i$$
$$= (n-i+1)(o_i-s_i) + (n-j+1)(s_i-o_i)$$
$$= (n-i+1)(o_i-s_i) - (n-j+1)(o_i-s_i)$$
$$= ((n-i+1)-(n-j+1))(o_i-s_i) = (j-i)(o_i-s_i)$$

Since $j > i$ & $o_i > s_i$, then the difference in cost is positive which contradicts the assumption that O was optimal.

As case 1 and 3 are contradictions and case 2 is our only viable solution, our proposed set where the zombie's time to reach the group being in ascending order is optimal.