Luis Enrique Franco Marín

410921353

Forecast Week 8: Interval Cover

1. Analyze worst case time complexity of above algorithm.

Given an array of intervals these most be sorted (lg n) and then the intervals must be analyzed one by one in the worst case scenario. So O(n lg n)

2. Analyze worst case time complexity of above algorithm with sorted V.

If the array is sorted and as is the worst case, all intervals must be analyzed so O(n)

3. Rewrite above algorithm to detect target range may impossible to be covered by original interval set. Analyze space complexity, time complexity of best case and worst case of modified algorithm.

1. Set current position p ← l ∗, answer set V ′ ← { }.

2. Find right-most (have largest right end) interval v ∈ V that l v ≤ p.

3. Update p ← r v , V ′ ← V ′ ∪ {v}.

4. Repeat step 2 and 3 until p ≥ r ∗

5. If largest right end p<r\* return -1

4. Above algorithm is doing through left to right. Please rewrite it to a through right to left version. Is your new algorithm always get the same result of original one? Give example(s) to explain your answer.

1. Set current position p ← r ∗, answer set V ′ ← { }.

2. Find left-most (have smallest left end) interval v ∈ V that r v ≥ p.

3. Update p ← l v , V ′ ← V ′ ∪ {v}.

4. Repeat step 2 and 3 until p ≤ l ∗

Example:

Original algorithm

Input: V = {{1, 3}, {2, 4}, {2, 10}, {2, 3}, {1, 1}}

Interval = {1, 10}

p←1, V’{}

p←3, V’{{1, 3}}

p←10 V’{{1, 3}, {2, 10}}

Right to left algorithm

Input: V = {{1, 3}, {2, 4}, {2, 10}, {2, 3}, {1, 1}}

Interval = {1, 10}

p←10, V’{}

p←2, V’{{2, 10}}

p←1 V’{{1, 3}, {2, 10}}