CSE 421

Approximation Alg

Shayan Oveis Gharan
An Idea

Choose a number $w$ from $x_1, \ldots, x_n$

Define

- $S_<(w) = \{x_i : x_i < w\}$
- $S_\leq(w) = \{x_i : x_i = w\}$
- $S_>(w) = \{x_i : x_i > w\}$

Solve the problem recursively as follows:

- If $k \leq |S_<(w)|$, output $Sel(S_<(w), k)$
- Else if $k \leq |S_<(w)| + |S_\leq(w)|$, output $w$
- Else output $Sel(S_>(w), k - |S_<(w)| - |S_\leq(w)|)$

Ideally want $|S_<(w)|, |S_>(w)| \leq n/2$. In this case ALG runs in $O(n) + O\left(\frac{n}{2}\right) + O\left(\frac{n}{4}\right) + \cdots + O(1) = O(n)$. Can be computed in linear time
Partition into $n/5$ sets. Sort each set and set $w = \text{Sel}(\text{midpoints}, n/10)$

- $|S_{<}(w)| \geq 3 \left( \frac{n}{10} \right) = \frac{3n}{10}$
- $|S_{>}(w)| \geq 3 \left( \frac{n}{10} \right) = \frac{3n}{10}$

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n) \Rightarrow T(n) = O(n)$$
Median Algorithm

\[ \text{Sel}(S, k) \{ \]
\[ n \leftarrow |S| \]
If \((n < ??)\) return ??
Partition \(S\) into \(n/5\) sets of size 5
Sort each set of size 5 and let \(M\) be the set of medians, so \(|M| = n/5\)
Let \(w = \text{Sel}(M, n/10)\)
For \(i = 1\) to \(n\)\{
  If \(x_i < w\) add \(x\) to \(S_<(w)\)
  If \(x_i > w\) add \(x\) to \(S_>(w)\)
  If \(x_i = w\) add \(x\) to \(S_\leq(w)\)
\}
If \((k \leq |S_<(w)|)\)
  return \(\text{Sel}(S_<(w), k)\)
else if \((k \leq |S_<(w)| + |S_\leq(w)|)\)
  return \(w\);
else
  return \(\text{Sel}(S_>(w), k - |S_<(w)| - |S_\leq(w)|)\)
\}

We can maintain each set in an array
Approximation Algorithms
How to deal with NP-complete Problem

Many of the important problems in real world are NP-complete.

SAT, Set Cover, Graph Coloring, TSP, Max IND Set, Vertex Cover, …

So, we cannot find optimum solutions in polynomial time. What to do instead?

• Find optimum solution of special cases (e.g., random inputs)

• Find near optimum solution in the worst case
Approximation Algorithm

Polynomial-time Algorithms with a guaranteed approximation ratio.

\[ \alpha = \frac{\text{Cost of computed solution}}{\text{Cost of the optimum}} \]

worst case over all instances.

Goal: For each NP-hard problem find an approximation algorithm with the best possible approximation ratio.
Given a graph $G=(V,E)$, Find smallest set of vertices touching every edge
Greedy algorithms are typically used in practice to find a (good) solution to NP-hard problems.

**Strategy (1):** Iteratively, include a vertex that covers most new edges.

Q: Does this give an optimum solution?
A: No,
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most
Greedy (1): Pick vertex that covers the most

Greedy Vertex cover = 20
OPT Vertex cover = 8
Greedy (1): Pick vertex that covers the most $n$ vertices. Each vertex has one edge into each $B_i$.

Each vertex in $B_i$ has $i$ edges to top.

Greedy pick bottom vertices $= n + \frac{n}{2} + \frac{n}{3} + \cdots + 1 \approx n \ln n$

OPT pick top vertices $= n$
A Different Greedy Rule

Greedy 2: Iteratively, pick both endpoints of an uncovered edge.

Vertex cover = 6
Greedy 2: Pick Both endpoints of an uncovered edge

Greedy vertex cover = 16

OPT vertex cover = 8
Greedy (2) gives 2-approximation

**Thm:** Size of greedy (2) vertex cover is at most twice as big as size of optimal cover

**Pf:** Suppose Greedy (2) picks endpoints of edges $e_1, \ldots, e_k$. Since these edges do not touch, every valid cover must pick one vertex from each of these edges!

i.e., $OPT \geq k$.

But the size of greedy cover is $2k$. So, Greedy is a 2-approximation.
Set Cover

Given a number of sets on a ground set of elements, the **Goal**: choose minimum number of sets that cover all.

For example, a company wants to hire employees with certain skills.
Set Cover

Given a number of sets on a ground set of elements,

Goal: choose minimum number of sets that cover all.

Set cover = 4
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered
A Greedy Algorithm

**Strategy**: Pick the set that maximizes # new elements covered
A Greedy Algorithm

**Strategy**: Pick the set that maximizes number of new elements covered.
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered

Thm: Greedy has $\ln n$ approximation ratio
A Tight Example for Greedy
A Tight Example for Greedy
A Tight Example for Greedy
A Tight Example for Greedy
A Tight Example for Greedy
A Tight Example for Greedy

Greedy = 5

OPT = 2
Thm: If the best solution has k sets, greedy finds at most k ln(n) sets.

Pf: Suppose OPT=k
There is set that covers 1/k fraction of remaining elements, since there are k sets that cover all remaining elements. So in each step, algorithm will cover 1/k fraction of remaining elements.

#elements uncovered after t steps 
\[ \leq n \left( 1 - \frac{1}{k} \right)^t \leq ne^{-\frac{t}{k}} \]
So after \( t = k \ln n \) steps, # uncovered elements < 1.