CS206M Notes

Gautam Singh
April 10, 2013

---

1 Recap of some standard problems

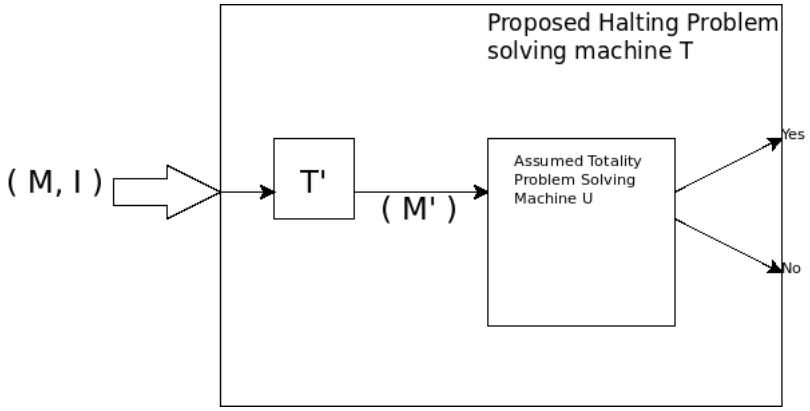
1.1 Halting Problem

Problem that answers the question whether a given machine described by M halts or runs forever on a given input I.

1.2 Totality Problem

Problem that answers the question whether a given machine described by M halts on every input or not.

2 Proving by reduction that Totality Problem is undecidable



- Scheme: We shall reduce halting problem to totality problem and say that solving totality problem solves halting problem. Hence totality problem must be as hard as the halting problem. Now since we know that halting problem is undecidable hence totality problem is also undecidable.
- Construction of Halting Problem machine is obvious from the diagram.
- (M,I) is the tuple (Machine Description, Input).
- Also, machine T' makes a machine M'and outputs its description.
- Machine M' works as follows:

  if input = I : simulate M
  else: halt

- Now machine M' will halt on every input only if machine M halts on input I. Hence we feed this new machine into totality problem solving machine U and tap the output as output of halting problem solving machine T.

3 Pseudo-polynomial time Algorithms

Algorithms having polynomial running times in terms of the numerical value of the input. In reality, since the numbers are represented using bits hence numerical value is exponential in terms of the actual bit-length of the input. So our polynomial time algorithms are really exponential in terms of word-length but we continue to call them polynomial time.

- Example: Knapsack

4 How to compare hardness of two problems

- Example: For strings, substring problem is atleast as hard as equality problem. This happens because if we can solve substring problem using algorithm A then we can convert the solution of algorithm A with O(1) overhead into an algorithm B which solves equality problem. In this case all we need to do is check whether length of 2 strings is same and whether the longest substring given by algorithm A has length equal to lengths of both the input strings. This overhead is clearly O(1).
- Example: For knapsack problem, optimisation version is atleast as hard as decision version. This happens because optimisation version directly gives a solution of decision version as well. But same cannot be said for vice-versa case. Hence the claim.

5 Heuristics

In algorithms of this domain, the programmer has no obligation to make an optimal solution. She/He may not even need to give a mathematical proof about accuracy or precision or completeness of that algorithm. Benefit is that we can solve the problem fast.

6 Approximation

Again the solution need not be an optimal solution but still the programmer can make mathematically provable claims about the optimality, precision etc. on an average.

6.1 Example: Knapsack Problem approximate solution

- Given: For a list of items we know value and volume of each of the item. Also we all we know about the knapsack is the volume. We want to put such a combination of items such that value of knapsack is optimal. But we would rather make an approximate approach here.
- Algorithm:
  - Remove all items having individual volumes greater than the knapsack volume. These cannot be fit in any case and are useless.
  - Now sort the items in increasing Value per Volume order.
  - Now start putting them into the knapsack one by one in order of decreasing density till $(i+1)^{th}$ item cannot be fit.
  - Now we shall call the volume of these i items W.
  - Now lets assume that we have a virtual knapsack with capacity W' such that $(i+1)$ items exactly fit.
  - Now we shall compare the combined value of first i items i.e. u and individual value of $(i+1)^{th}$ item i.e. v.
  - Now depending on whether u or v is greater we shall either carry first i items or just $(i+1)^{th}$ item.
- Calculating bound on optimality
  - We claim that $v+u$ i.e. combined value of first $i+1$ items is the optimal value for knapsack volume W' which we denote by $op(W')$.
  - Next we claim that $op(W) \leq op(W')$.
  - Note that both the above claims can be proved.
  - Now we can infer from mathematics that for any two values x and y, $max(x,y) \geq (x+y)/2$. Hence,

$$max(u,v) \geq (v+u)/2 = op(W')/2 \geq op(W)/2$$

    hence,

$$max(u,v) \geq op(W)/2$$

  - Since we know that $max(u,v)$ is our approximate solution, hence we have a bound on the optimality of our approximate solution.

END