Design and Analysis of Algorithms

L11: Algo Design Ideas

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Resources

- Martin Richards
  - Notes on Data Structures and Algorithms
- Text book 2: Horowitz
- Text book 1: Levitin
- https://visualgo.net/en

Ideas on Algorithm Design

- Recognize a variant on known problem
- Reduce to a simpler problem
- Divide and Conquer
- Estimation of cost by recurrence relation
- Dynamic Programming
- Greedy Algorithm
- Back tracking
- Hill Climbing
- Identify wasted work in a simple method
- Find a mathematical lower bound
- Million Monkey method

Variant on a Known Problem

- Try to identify a related problem whose solution is known
- Use solution of one problem to solve essentially tricky part of other problem.
- Solve the problem in totality
- Example 1: multiply two numbers
  - Known variant: add two numbers
  - How to use addition of two numbers to achieve multiplication
- Example 2: Class assignment submission
  - variant: Identify the classmate who has done it
  - Copy the assignment with change of USN

Reduction to a Simpler Problem

- Typically works where induction methods are used to show correctness proof
- Recursive functions take this appproach
- Example: Compute n^2

  $n^2 = (n-1)^2 + 2(n-1)$
- Example: factorial `n`

  `n! = n*(n-1)!`
- Example: GCD computation

  `gcd(x,y)= gcd(r,x)` where `y=kx+r`

Divide and Conquer Algo

- Divide (break) the problem (size `n`) into similar sub problems
  - Size of sub problems should be some factor of original e.g. `n/c`
    - When small enough, solve by brute force
- Conquer (Solve) the sub-problem
  - Use recursion to solve small problem
- Combine (Merge) the solution of sub-parts
- The cost is
  - cost of breaking
  - cost of solving subproblem
  - cost of combining

Cost Estimation by Recurrence

- Use recurrence relation to find the cost of operation
  - Once the cost is known, then algorithm framework is worked out
  - Design the algorithm on how to break the given problem
- Example: Mergesort

  `f(n) = 2f(n/2)+n`

  `-> f(n) = O(nlog`$_2$`n)`
- Example: Word search in English dictionary

  `f(n) = f(n/k) + c`

  `-> f(n) = log`$_k$`n`

Dynamic Programming

- Build a table of solutions to smaller versions of the problem
- Work towards the solution by using this table
- Example: Binomial Coefficient

```
    1          (a+b)0
   1 1         (a+b)1
  1 2 1        (a+b)2
 1 3 3 1       (a+b)3
1 4 6 4 1      (a+b)4
```

- Example: Computing `n2`

```
        1            12
      1+2+1           22
    1+2+3+2+1          32
  1+2+3+4+3+2+1         42
1+2+3+4+5+4+3+2+1        52
```

Greedy Algorithms

- Useful when some sort of optimizations is involved.
- Basic idea:
  - Perform whatever operation contributes most towards the final goal
  - Next step will be same approach after the previous step
  - Note: Final solution may not always be optimal
- Example:
  - Eating food in restaurant: get whatever hot now
    - No waiting is to be done
  - Taking exam (you know all answers) but time is less
    - Which questions to start writing

Backtracking

- When algorithm involves search (explore)
  - backtracking is useful
- Split the search procedure in multiple paths(parts)
  - Start with one path
  - If solution not find and path ends,
  - Backtrack to previous starting point and
    - search on next path
- Example:
  - Finding a solution to the maze (childhood comicbooks)
  - Chessboard: queens placement
  - Chessboard: Knights traversal

Hill Climbing

- Generally, used in optimization problems
- Start with some feasible (non-optimal) solution
- Incrementally work towards improving the solution
- May not always find the optimal solution
- Analogy: Climbing to hilltop in hilly region
- Example:
  - Getting a job(placement)
    - Start with some placement (job)
    - Work towards a better placement (better company)

Identify Wasted Work

- Design a simple solution to the problem
- Analyze the solution and
  - identify critical costly part of the solution
- Attack the weakness of critical part to improve the solution
- Example (real life):
  - Phone converstation, and you are put on hold
  - can do some useful work while on call hold
- Example (CS): CPU Scheduling
  - when a processing is waiting for I/O
  - other process is assigned CPU

Seek Mathematical Lower Bound

- Establish a proof that some task must take at least certain minimum time
- Use this insight to design the algorithm
- A properly proved lower bound can prevent wasted time seeking improvement
- Example: Sorting `n` numbers:
  – min time: `ceil(log`$_2$`n!)` comparisons
  – Consider `n=5`, min time = `log`$_2$`(120)=7`
  – Design the algorithm

Million Monkey (MM) Method

- Problem: Give one typewriter each to million monkeys (random character press) and a milliion years, and one of the monkey will write the Shakespear play.
  - Idea: give a problem to a group of researchers and sufficient time, they will be able to find the solution of the problem.
- Example: expected output '`hello`'.
  - Assume typewriter has 50 keys
  - Probability of not typing '`hello`' = $1-(1/50)^5$
  - Probability of not typing hello by million monkeys in 10000 tries = $(1-(1/50)^5)^{10} = 1.2*10^{-14}$
  - Prob. of typing hello = $1-1.2*10^{-14}=1$

Fun Exercise of Game of 128 numbers

- A practical fun example of Data structures and Algorithm

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128

Game:
. Go thru a set of cards
. Say Y/N if present or not
. You will get you number graphically displayed to you

Q?:
Which algorithm we are discussing?
Aim: Can we find more such examples

Game of 128 numbers - b

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

x

Exercise G

- Exercise G
  - Work out the remaining 6 cards

Summary

- Ideas for algorithm design
  - Recognize a variant on known problem
  - Reduce to a simpler problem
  - Divide and Conquer
  - Estimation of cost by recurrence relation
  - Dynamic Programming
  - Greedy Algorithm
  - Back tracking
  - Hill Climbing
  - Identify wasted work in a simple method
  - Find a mathematical lower bound
  - Million Monkey method