Introduction to ADS

Data Structure

- Balanced Search Trees
    - AVL Trees/Splay Trees **with Amortized Analysis**
    - Red Black Trees
    - B+ Trees

> - How to make a search tree balanced?
> - How to maintain a balanced search tree?
> - How much time is required for search, insertion, and deletion?worst-case? average-case?

- Heaps
    - lefist Heaps/Skew Heaps
    - Binomial Queue

> - Find-min, insertion, deletion-min
> - How to merge two heaps efficiently?

- Inverted File Index*

Computational Problem

> A Bit More Detials
>
> Or "[61B SP24] Lecture 39 - Computability"

3 Problems

以 Knapsack Problem 为例

- Question1: Given a list of items and a weight limit, return the list of items you should steal
- Question2: Given a list of items and a weight limit, return the most value of items you could steal
- Question3: Given a list of items, a weight limit, and a target value, return True if you can steal that amount of stuff or more, and False if you can't

> Question1是 **Optimization Problems: finding an optimal solution**
>
> Question2是 **Optimization Problems: Computing the optimal value**
>
> Question3是 **Decision Problems(Or Yes/No Problem)**

Lemma: 这三个问题可以归约(reduction)为一个等价问题

> 此处等价是指：给足相应细节的条件，三个问题中如果有个问题能在constant time被解决，那么另外的问题必然能通过某个算法在Poly-time(多项式时间)内解决

proof:

$3 \rightarrow 2:$

```
GUESS = 0;
while(True){
    ASK(GUESS) ? break; GUESS++;
}
```

GUESS有上界，所以该算法必有解且正确

$2 \rightarrow 1:$

```
most_value = MOST_VALUE(items)
add all item in items into LIST
for(item in items){
    rest_max_value = MOST_VALUE(the_rest_items) //Answer of 2
    if(rest_max_value == most__value){
        remove item from the LIST
    }
}
```

$1 \rightarrow 3$等

略. □

> If we have a solution to any one of these three problems, we can create a solution to the other two. Thus, we can consider these three problems to be equivalent under Turing Reduction.

Something about P，NP......

Tips: 接下来的表述纯个人理解，想看更严谨的表达，请看开头提供的网址

- DTM(Deterministic Turing Machine): 确定性图灵机，即任何一个能用图灵完备语言实现的，返回值为Boolean类型的函数。我们定义P为能用 DTM 在 Poly-Time 解决的 `Decision Problem` 所构成的集合
- NTM(Nondeterministic Turing Machine)则是可以给出猜测,可以理解为函数能返回一个 `switch` 表中的值.如果有一个值能使得 `Decision Problem` 的答案为真，那么这个 NTM 也会返回真;否则返回假. NP 是能用 NTM 在 Poly-Time解决的 `Decision Peoblem` 所构成的集合

一般来说，NTM解决问题有两步

- 生成一个猜测
- 确定这个猜测是否符合要求

> Because of this, NP is also defined as the set of problems whose solution can be verified in polynomial time by a DTM.Contrast to P, which is the set of problems whose solutions can be generated in polynomial time

当然对于NP问题，能不能高效验证猜测同样十分重要

> We will call a problem NP-Hard if every problem in NP reduces to that problem (or in other words, that problem is at least as hard as every problem in NP)

SAT: 给定一个关于布尔变量（例如 "x && y", "x && (!x)"）的函数，是否存在一种为所有变量赋值的方式，使得该布尔表达式的最终计算结果为 True（真）？

> SAT is NP-Hard. In addition, SAT is in NP. We call a problem that's both in NP and NP-Hard an NP-Complete problem. Any NP-Complete problem is the hardest problem in NP.

Algorithms

算法衡量标准

> We can think of an algorithm with linear or nearlinear running time as a primitive that we can use essentially “for free,” since the amount of computation used is barely more than what is required just to read the input. -- *《算法详解 I》*

- decision problems
    - Correctness and Efficiency
- Optimized problems
    - Trade-off between Cost(Efficiency) and Quality(Effectivity)

Exact Algorithms

- Divide and Conquer
- Backtracking
- Dynamic Programming

Heuristic Algorithms

- Greedy Algorithms
- Local Search

Approximation Algorithms

> NP - complement 为前提

- Approximation Algorithms
- Randomized Algorithms

Others

- Parrallel Algorithms
- External Algorithms