R19 Complexity

0-1 Knapsack Revisited

- 0-1 Knapsack
  -Input: Knapsack with volume S, want to fill with items: item i has size si and value vi.
  -output: Output: A subset of items (may take 0 or 1 of each) with $\sum s_i$ ≤ S maximizing $\sum v_i$
  -Solvable in O(nS) time via dynamic programming
- How does running time compare to input?
  -What is size of input? If numbers written in binary, input has size O(n log S) bits。 *n numbers integers <= S*
  -Then O(nS) runs in expoential time compared to the input
  -If numbers polynomially bounded, S = $n^{O(1)}$, then dynamic program is polynomial
  -*This is called a pseudopolynomial time algorithm*
- Is 0-1 Knapsack solvable in polynomial time when numbers not polynomially bounded?
  No if P != NP.

Decision Problems

- Decision Problem
  assignment of inputs to NO (0) or YES (1).
- Inputs are either No instances or Yes instances (i.e. satisfying instances)
- Algorithm/Program
  constant length code (working on a word-RAM with Ω(log n)-bit words) to solve a problem, i.e., it produces correct output for every input and the length of the code is independent of the instance size
- Problem is decidable if there exists a program to solve the problem in finite time

Decidability

- Program is finite string of bits, problem is function p : N → {0, 1}, i.e. infinite string of bits
- Proves that most decision problems not solvable by any program
- e.g. the Halting problem is undecidable
- Fortunately most problems we think of are algorithmic in structure and are decidable

Decidable Problem Classes

R	problems decidable in finite time	'R' comes from recursive languages
EXP	problems decidable in exponential time $2^{n^{O(1)}}$	most problems we think of are here
P	problems decidable in polynomial time $n^{O(1)}$	efficient algorithms, the focus of this class

- These sets are distinct, i.e. $\mathbf{P} \subsetneq \mathbf{EXP} \subsetneq \mathbf{R}$ (via time hierarchy theorems, see 6.045)

Nondeterministic Polynomical Time(NP)

- P is the set of decision problems for which there is an algorithm A such that for every instance I of size n, A on I runs in poly(n) time and solves I correctly
- NP is the set of decision problems for which there is an algorithm V , a "verifier", that takes as input an instance I of the problem, and a "certificate" bit string of length polynomial in the size of I, so that:
  -V always runs in time polynomial in the size of I,
  -if I is a YES-instance, then there is some certificate c so that V on input (I,c) returns YES, and
  -if I is a NO-instance, then no matter waht c is given to V together with I, V will always output NO on (I,c).
- You can think of the certificate as a proof that I is a YES-instance. If I is actually a NO instance then no proof should work.

Problem	Certificate	Verifier
s-t Shortest Path	A path P from s to t	Adds the weights on P and checks if $\le d$
Negative Cycle	A cycle C	Adds the weights on C and checks if < 0
Longest Path	A path P	Checks if P is a **simple** path with weight at least d
Subset Sum	A set of items A'	Checks if $A' \in A$ has sum S
Tetris	Sequence of moves	Checks that the moves allow survival

- $P \in NP$ if you can solve the problem, the solution is a certificate
- Open: Does P = NP? NP = EXP?
- Why do we care? If can show a problem is hardest problem in NP, then problem cannot be solved in polynomial time if P != NP

Reductions

A input -> B input ---> B solution -> A solution

A	Conversion	B
Unweighted Shortest Path	Give equal weights	Weighted Shortest Path
Product Weighted Shortest Path	Logarithms	Sum Weighted Shortest Path
Sum Weighted Shortest Path	Exponents	Product Weighted Shortest Path

- Problem A is NP-Hard if every problem in NP is polynomially reducible to A
- i.e. A is at least as hard as (can be used to solve) every problem in NP ($X \leq A$ for $X \in NP$)
- *NP-Complete = NP and NP-Hard*
- All NP-Complete problems are equivalent, i.e. reducible to each other
- First NP-Complete? Every decision problem reducible to satisfying a logical circuit.
- Longest Path, Tetris are NP-Complete, Chess is EXP-Complete