Chris Bero

HW 4 CS317

1. Dynamic Programming Solves multiple subproblems and builds up a final answer. DP works by solving small subproblems and placing their results in a table, then solving larger subproblems and doing the same, until the final result is reached.
   - Characterize the structure of an optimal solution.
   - Recursively define Values of optimal Solutions.
   - Compute the Value of an optimal Solution in a bottom-up fasion with a table.
   - Express the final solution in terms of optimal sub-solutions.

   - Optimal Substructure: The optimal Way to put parenthesis on the subchain $(A_1 \cdots A_K)$ With the Same for $A_{K+1} \cdots A_n$ must be the optimal Way for $A_1 \cdots A_n$.
   - Overlapping Subproblems: DP often results in many subproblems being Worked repeatedly. This is handled by using an auxiliary table to store sub-results.

4. Floyd's algorithm is a DP technique which can handle negative values, and is used instead of running Dijkstra's on each vertex.

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 2 & \infty & 1 & 8 \\ B & 6 & 0 & 3 & 2 & \infty \\ C & \infty & \infty & 0 & 4 & \infty \\ D & \infty & \infty & 2 & 0 & 3 \\ E & 3 & \infty & \infty & \infty & 0 \end{array} \sim \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 2 & \infty & 1 & 8 \\ B & 6 & 0 & 3 & 2 & 14 \\ C & \infty & \infty & 0 & 4 & \infty \\ D & \infty & \infty & 2 & 0 & 3 \\ E & 3 & 5 & \infty & 4 & 0 \end{array} \sim \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 2 & 5 & 1 & 8 \\ B & 6 & 0 & 3 & 2 & 14 \\ C & \infty & \infty & 0 & 4 & \infty \\ D & \infty & \infty & 2 & 0 & 3 \\ E & 3 & 5 & 8 & 4 & 0 \end{array}$$

$$\begin{array}{c|ccccc} A & 0 & 2 & 3 & 1 & 4 \\ B & 6 & 0 & 3 & 2 & 5 \\ C & \infty & \infty & 0 & 4 & \infty \\ D & \infty & \infty & 2 & 0 & 3 \\ E & 3 & 5 & 6 & 4 & 0 \end{array} \quad \begin{array}{c|ccccc} A & 0 & 2 & 3 & 1 & 4 \\ B & 6 & 0 & 3 & 2 & 5 \\ C & \infty & \infty & 0 & 4 & \infty \\ D & 6 & 8 & 2 & 0 & 3 \\ E & 3 & 5 & 6 & 4 & 0 \end{array}$$

Chris Dero

HW 4 CS 317

2. Matrix Chain Multiplication involves taking a sequence of matrices $(A_1, A_2, \ldots, A_n)$ and computing the product $A_1 A_2 \cdots A_n$. The problem is to determine the most efficient method of associative grouping to reduce scalar multiplications.

1. Determine the structure of an optimal parenthesization.
2. Define the cost of an optimal solution recursively in terms of the optimal solutions to subproblems.
3. Compute optimal costs with a tabular, bottom-up approach.
4. Construct an optimal solution from the returned table.

3. In the Longest Common Subsequence problem, we derive the maximum length common subsequence from two sequences.

1. Characterize a longest common subsequence.
2. Establish a recurrence for the value of an optimal solution.
3. Use Dynamic Programming to compute solutions bottom-up.
4. Construct an LCS from the returned table.

3.

|  | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 ↑ | 1↖ | 1← | 1↖ | 1↖ | 1← | 1↖ | 1↖ | 1← |
| 0 | 1↖ | 1← | 2↖ | 2← | 2← | 2↖ | 2← | 2← | 2↖ |
| 0 | 1↖ | 1← | 2↖ | 2← | 2← | 3↖ | 3← | 3← | 3↖ |
| 1 | 1↑ | 2↖ | 2← | 3↖ | 3↖ | 3← | 4↖ | 4↖ | 4← |
| 0 | 1↖ | 2↑ | 3↖ | 3← | 3← | 4↖ | 4← | 4← | 5↖ |
| 1 | 1↑ | 2↖ | 3↑ | 4↖ | 4↖ | 4← | 5↖ | 5↖ | 5← |
| 0 | 1↖ | 1← | 3↖ | 4↑ | 4↑ | 5↖ | 5← | 5← | 6↖ |
| 1 | 1↑ | 2↖ | 3↑ | 4↖ | 5↖ | 5← | 6↖ | 6↖ | 6← |

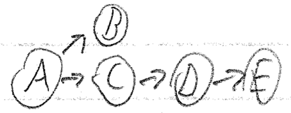
LCS: 010101

5. 0-1 Knapsack entails finding the optimal use of a Carrying devices capacity. With DP, we build a solution by starting with an empty set and adding items to our recurrance relation.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 25 | 25 | 25 | 25 |
| 2 | 0 | 0 | 20 | 20 | 20 | 45 | 45 |
| 3 | 0 | 15 | 15 | 35 | 35 | 35 | 60 |
| 4 | 0 | 15 | 15 | 35 | 40 | 55 | 60 |
| 5 | 0 | 15 | 15 | 35 | 40 | 55 | 65 |

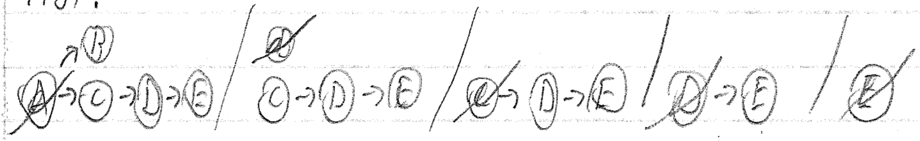
Max Value: $65
items: 5, 3

6. Depth First Search explores as far down one branch as possible before moving to the next, commonly uses a stack to track progress.



Pop Order: E, D, C, B, A  Sorted: A, B, C, D, E

Source Removal visits a vertex, and then "removes" it from the graph to be added to the traversal list.



Topological Order: A, B, C, D, E

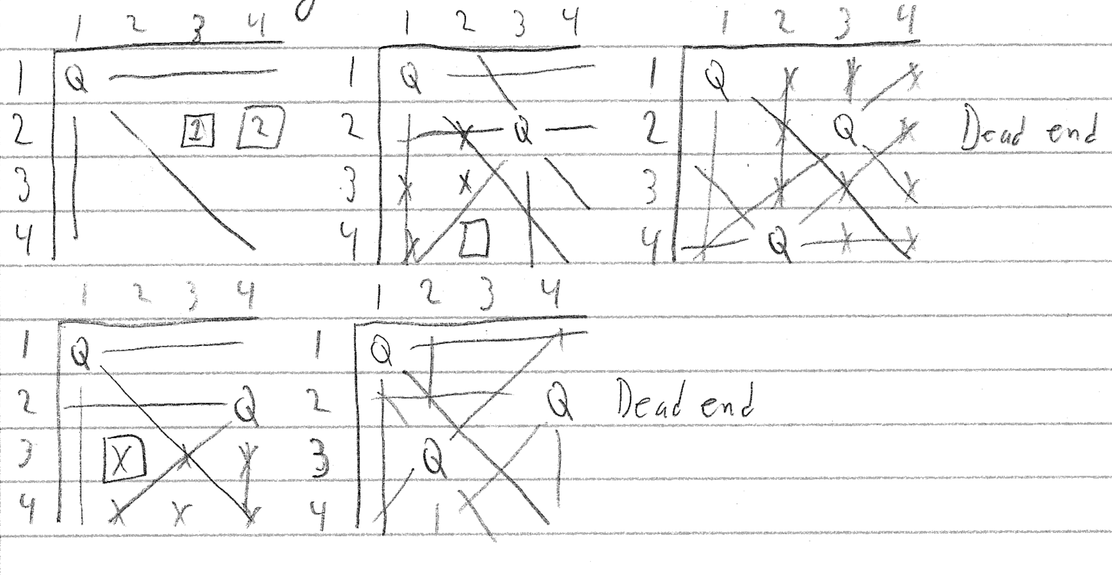
Chris Bero

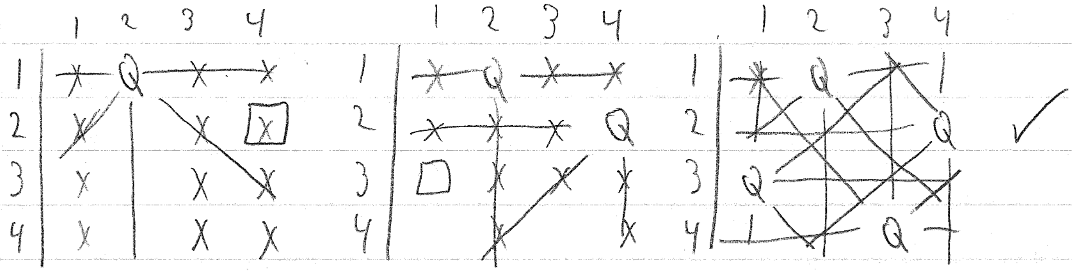
HW 4 CS 317

7. P: The class of decision problems that are solvable in O(p(n)) time, where p(n) is a polynomial of input size.
Np: The class of decision problems whose proposed solutions can be verified in polynomial time; solvable by a nondeterministic polynomial algorithm.
Np-Complete: A decision problem is Np-Complete if it's as hard as any problem in NP.

If a P algorithm is found for an NP-Complete problem, then P=NP, currently this isn't the case. We study these in order to develop alternate algorithms which provide somewhat better efficiency.

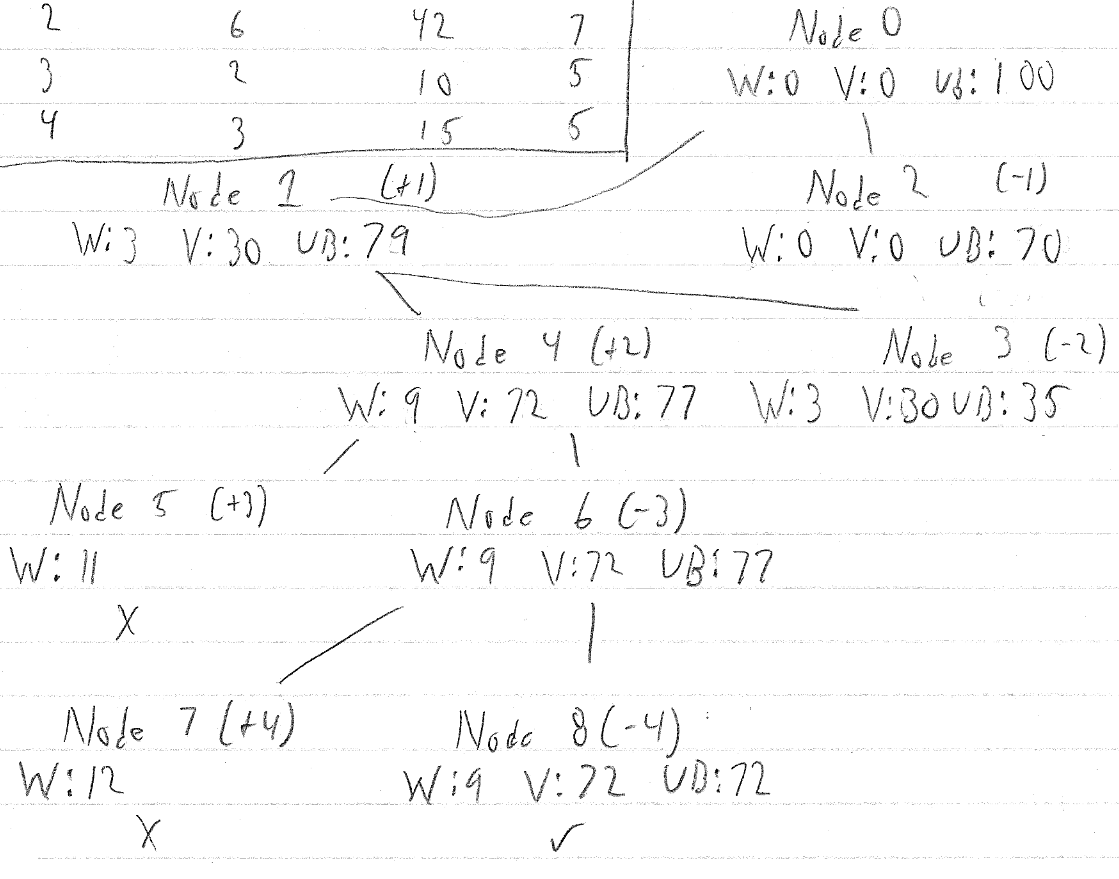
8. Backtracking is a better variation of exhaustive enumeration, it constructs solutions one piece at a time and evaluates partial solutions. n-Queens is a problem solvable with backtracking, which involves placing n number of Queens on an nxn chessboard such that they cannot conflict.





9. Branch and Bound is a general algorithm for finding optimal solutions of various optimization problems. Requires a state-space tree for every node, and the best solution's value.

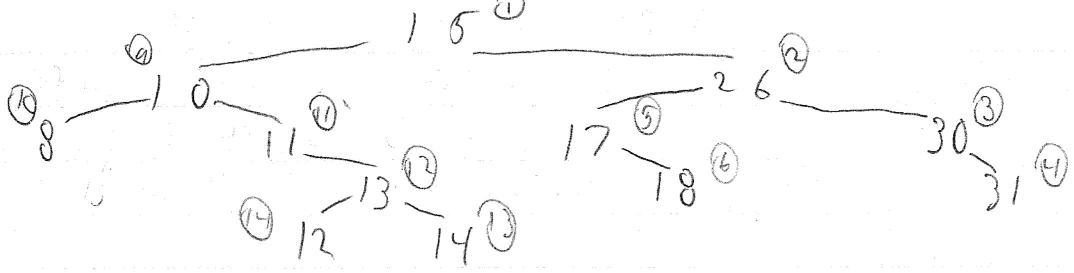
| Item | Weight | Value | V/W |
|---|---|---|---|
| 1 | 3 | 30 | 10 |
| 2 | 6 | 42 | 7 |
| 3 | 2 | 10 | 5 |
| 4 | 3 | 15 | 5 |



Chris Ben

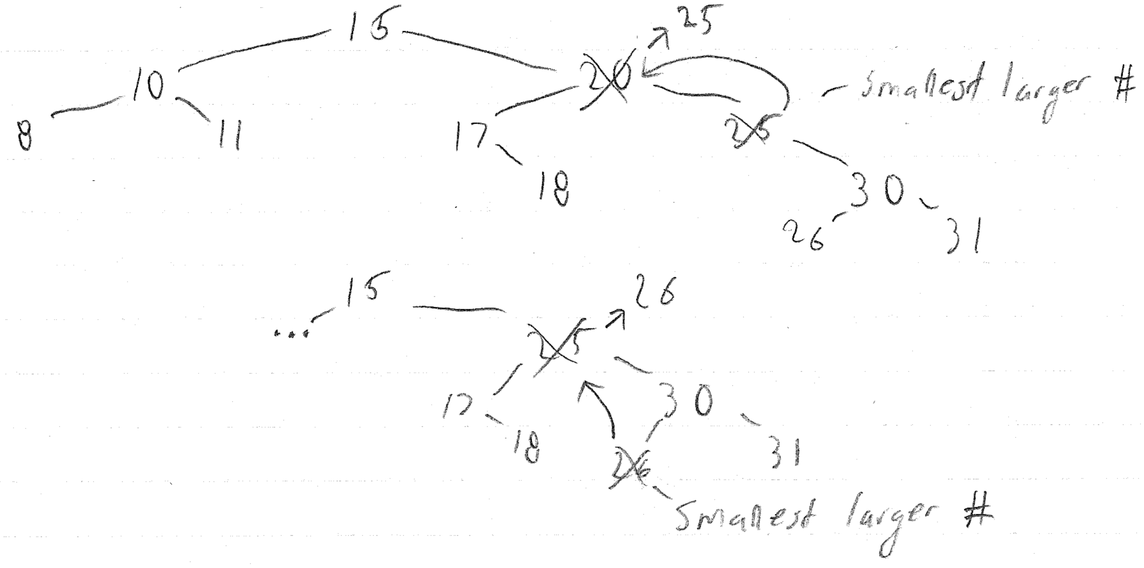
HW4 CS 317

10.



Worste case efficiency occurs with pre-sorted lists.
Such as: 8, 10, 11, 12, 13, 14, 15, 17, 18, 26, 30, 31

11.



12. Huffman Coding gives priority to more frequently used characters and asigns them Shorter binary Values to compress communications.

~~Honorificabilitudinitatibus~~ 27

S:1, D:1, L:1, C:1, F:1, R:1, H:1, U:2, B:2, A:2, N:2, O:2, T:3, I:7

H:1, U:2, B:2, A:2, N:2, O:2, T:3, I:7, SD:2, LC:2, FR:2

T:3, I:7, SD:2, LC:2, FR: 2, HU:3, BA:4, NO:4

FR:2, HU:3, BA:4, NO:4, TI:10, SDLC:4

TI:10, SDLC:4, FRHU:5, BANO:8

TISDLC:14, FRHUBANO:13

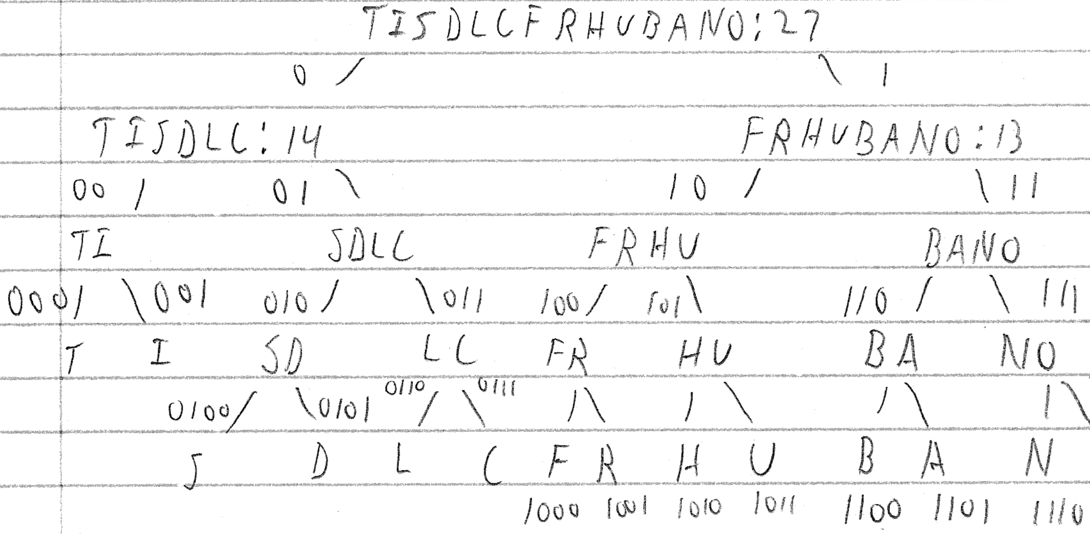
TISDLCFRHUBANO:27

H: 1
O: 2
N: 2
R: 1
I: 7
F: 1
C: 1
A: 2
B: 2
L: 1
T: 3
D: 1
U: 2
S: 1

Chris Dero

HW4 CS317

12.



Encoded: 1010, 1111, 1110, 1111, 1001, 001, 1000, 001, 0111,

1101, 1100, 001, 0110, 001, 000, 1011, 0101, 001,

1110, 001, 000, 1101, 000, 001, 1100, 1011, 0100

Unencoded: 216 bits Encoded: 68+30 = 98

Ratio: 0.45