A* Search

Consider the "eight puzzle". There are eight tiles numbered 1 through 8 on a 3 by three grid with nine locations so that one location is left empty. We can "move" by sliding a tile adjacent to the empty location into the empty location. The goal is to take a given configuration and through a series of moves arrange the tiles in order from left to right and top to bottom. The set of configurations of the tiles forms a graph where two nodes (configurations) are connected if one node can be converted to the other by a single move.

We are interested in computing the shortest path from a given start node to a given goal node.

1 Dijkstra Shortest Path

We are given a graph G where each edge is assigned a non-negative weight and we are also given a start node s and goal node g. We want the path of least weight from s to g.

Procedure *Dijkstra Shortest Path*

1. Initialize $\mathcal{Q}$ to be the set containing the single pair $\langle s, 0\rangle$.
2. $\mathcal{S} \leftarrow \emptyset$
3. **while** $\mathcal{Q}$ is not empty and $\mathcal{S}$ does not contain an assignment to g:
4. Remove a pair $\langle n, w\rangle$ from Q with minimum weight w (over all elements of Q).
5. **if** S does not already contains a weight for n, i.e., if S does not contain any pair of the form $\langle n, w'\rangle$, then:
6. add $\langle n, w\rangle$ to S.
7. For each edge $\langle n, m\rangle$ with weight w' in the graph G add the pair $\langle m, w + w'\rangle$ to Q.
8. If $\mathcal{S}$ contains an assignment to g then terminate with success (a path has been found) else terminate with failure (there is no path).

Note that if removing $\langle n, w\rangle$ form Q results in adding $\langle m, w'\rangle$ then because edge weights are non-negative we have that $w' \geq w$. This implies that the weights removed from Q monotonically increase. This implies that when a pair $\langle n, w\rangle$ is added to S, w is the weight of the shortest path to n.

2 Admissible Heuristics

Let h be a function form the nodes of G to non-negative real numbers. The function h is admissible if, for any node n, the distance from n to g is at least $h(n)$. The function h is called *monotone* if for any edge $\langle n, m\rangle$ with weight w we have $|h(m) - h(n)| \leq w$ Suppose h is monotone and $h(g) = 0$. Consider a path from g to n. As we cross an edge moving from g to n along this path the increase in h can be no larger than the weight of the edge. So $h(n)$ cannot be larger than the weight of the path. This gives us that if $h(g) = 0$ and h is monotone then h is admissible. Manhattan distance in the eight puzzle is both monotone and admissible (this is the sum over tiles of the number of rows plus the number of columns that the tile must be moved).

3 A*

The following differs from Dijkstra shortest path only in the priority used in step 4.

Procedure *A**

1. Initialize $\mathcal{Q}$ to be the set containing the single pair $\langle s, 0\rangle$.
2. $\mathcal{S} \leftarrow \emptyset$
3. **while** $\mathcal{Q}$ is not empty and $\mathcal{S}$ does not contain an assignment to g:
4. Remove a pair $\langle n, w\rangle$ from Q minimizing $w + h(n)$ (over all elements of Q).
5. **if** S does not already contains a weight for n, i.e., if S does not contain any pair of the form $\langle n, w'\rangle$, then:
6. add $\langle n, w\rangle$ to S.
7. For each edge $\langle n, m\rangle$ with weight w' in the graph G add the pair $\langle m, w + w'\rangle$ to Q.
8. If $\mathcal{S}$ contains an assignment to g then terminate with success (a path has been found) else terminate with failure (there is no path).

If h is monotone then the sum $w + h(n)$ of the item removed from the queue monotonically increases. This implies that when we remove $\langle n, w\rangle$ from the queue, the weight w is the least possible weight for n. Note that monotonicity alone gives correctness of the algorithm — we do not need that $h(g) = 0$.

By taking the distance to goal into account A* can be vastly more efficient than Dijkstra Shortest path.

4 A Star Parsing

A CFG for English might generate the phrase "that grand old house on the hill" using the following grammar.

$$
\begin{aligned}
NP &\rightarrow Det\ \ N \\
Det &\rightarrow \text{that} \\
Det &\rightarrow \text{the} \\
N &\rightarrow AP\ \ N \\
AP &\rightarrow \text{grand} \\
AP &\rightarrow \text{old} \\
N &\rightarrow N\ \ PP \\
N &\rightarrow \text{house} \\
PP &\rightarrow P\ \ NP \\
P &\rightarrow \text{on} \\
N &\rightarrow \text{hill}
\end{aligned}
$$

But the parsing accuracy of PCFGs can be improved by lexicalizing the nonterminals. This means a nonterminal is specified by a pair of a syntactic category and a particular word. For example NP_{house} is a nonterminal which can generate an noun phrase whose head word is the word "house". The concept of "head word" is not formally defined, but intuitively the head word is the "most important word" in a phrase. In a lexicalized PCFG the phrase "that grand old house on the hill" might be generated using the following productions.

$$
\begin{aligned}
NP_{\text{house}} &\rightarrow Det_{\text{that}} N_{\text{house}} \\
Det_{\text{that}} &\rightarrow \text{that} \\
N_{\text{house}} &\rightarrow AP_{\text{grand}} N_{\text{house}} \\
AP_{\text{grand}} &\rightarrow \text{grand} \\
N_{\text{house}} &\rightarrow AP_{\text{old}} N_{\text{house}} \\
AP_{\text{old}} &\rightarrow \text{old} \\
N_{\text{house}} &\rightarrow N_{\text{house}} PP_{\text{on}} \\
N_{\text{house}} &\rightarrow \text{house} \\
&\vdots
\end{aligned}
$$

Parsing a lexicalized PCFG is challenging. Even if we restrict the head words to the words that occur in the sentence, the average sentence length in the New York times is 25 words. Combining that with 20 syntactic categories gives 460 nonterminals. The viterbi algorithm considers all possible pairs of adgajent phrases — this approximately 25^3 times 460^2 — about 4 billion phrase pairs. Many parsers use even more information in the nonterminals making complete Viterbi parsing impossible.

5 Probabilities to Weights

Consider a probabilistic context free grammar (PCFG) in Chomsky normal form. We have have productions of the form $X \rightarrow YZ$ and $X \rightarrow a$ and for each nonterminal X we have $\sum_\beta P(X \rightarrow \beta) = 1$. We define the weight a production as follows.

$$w(X \rightarrow \beta) = \log_2 \frac{1}{P(X \rightarrow \beta)}$$

We can think of $w(X \rightarrow \beta)$ as a number of bits. The weight of a parse tree is the sum of the weights of the productions used in that parse tree. The weight of a tree can be thought of as the number of bits it takes to name (or code for) that tree. We are interested in finding the Viterbi parse tree — the most probably tree, or equivalently, the lightest weight tree.

6 Knuth Lightest Derivation

The efficiency of Parsing can be improved by using algorithms similar to Dijkstra shortest path and A*. Instead of "nodes" we work with "phrases" where a phrase is defined to be a triple $\langle X, i, j\rangle$ representing the generation of the string $a_i \dots a_{j-1}$ from the nonterminal X. We consider an input string $a_1 \dots a_n$. We are seking a derivation of the "goal phrase" $\langle S, 1, n+1\rangle$. In the following allgorithm the set $\mathcal{S}$ is often called the "chart" and algorithms of this form are often called chart parses. The essence of a chart parser is a table (the set $\mathcal{S}$) which stores dynamic programming values. Please compare the following algorithm to Dijkstra Shortest Path.

Procedure *Knuth Lightest Derivation for Parsing*

1.
2. Initialize Q to be the set containing all pairs $\langle\langle X, i, i+1\rangle\rangle$ where the grammar contains $X \rightarrow a$ with weight w and the ith word in the input string is a.
3. $\mathcal{S} \leftarrow \emptyset$
4. **while** $\mathcal{Q}$ is not empty and $\mathcal{S}$ does not contain an assignment to $\langle S, 1, n+1\rangle$:
5. Remove a pair $\langle\langle Y, i, j\rangle, w\rangle$ from Q minimizing w (over all elements of Q).
6. **if** S does not already contains a weight for $\langle Y, i, j\rangle$ then:
7. add $\langle n, w\rangle$ to S.
8. For each pair of the from $\langle\langle Z, j, k\rangle, w'\rangle$ in $\mathcal{S}$ where the grammar G conatins the production $X \stackrel{w''}{\rightarrow} YZ$ add the pair $\langle\langle X, i, k\rangle, w + w' + w''\rangle$ to $\mathcal{Q}$.
9. For each pair of the from $\langle\langle Z, k, j\rangle, w'\rangle$ in $\mathcal{S}$ where the grammar G conatins the production $X \stackrel{w''}{\rightarrow} ZY$ add the pair $\langle\langle X, k, j\rangle, w + w' + w''\rangle$ to $\mathcal{Q}$.
10. If $\mathcal{S}$ contains an assignment to $\langle S, 1, n+1\rangle$ then terminate with success (a parse has been found) else terminate with failure (there is no parse).

Note that if removing $\langle Y_{i,j}, w\rangle$ form Q results in adding $\langle Xu, v, w'\rangle$ then, because weights of grammar rules are non-negative, we have that $w' \geq w$. This implies that the weights removed from Q monotonically increase. This implies that when a pair $\langle X_{u,v}, w\rangle$ is added to S, w is the weight of the lightest weight derivation of $X_{u,v}$. Note that if the given word string has a light weight parse tree then heavy weight phrases are never considered.

7 A* Parsing

A context for phrase $\langle X, i, j\rangle$ (relative to input string $a_1 \ldots a_n$) is is a parse tree whose leaf nodes are labeled with $a_1 \ldots a_{i-1} X a_j \ldots a_n$. The weight of a context is the sum of the weights of the productions appearing in the parse tree. A function h with $h(\langle X, i, j\rangle)$ a real number will be called an admissible heuristic for parsing if $h(\langle X, i, j\rangle)$ is a lower bound on the weight of any context for $\langle X, i, j\rangle$. For a heuristic function h we define the following A* algorithm which differes from the preceding algorithm in step 4.

Procedure *A* Parsing*

1.
2. Initialize Q to be the set containing all pairs $\langle\langle X, i, i+1\rangle\rangle$ where the grammar contains $X \rightarrow a$ with weight w and the ith word in the input string is a.
3. $\mathcal{S} \leftarrow \emptyset$
4. **while** $\mathcal{Q}$ is not empty and $\mathcal{S}$ does not contain an assignment to $\langle S, 1, n+1\rangle$:
5. Remove a pair $\langle\langle Y, i, j\rangle, w\rangle$ from Q minimizing $w + h(\langle Y, i, j\rangle)$.
6. **if** S does not already contains a weight for $\langle Y, i, j\rangle$ then:
7. add $\langle n, w\rangle$ to S.
8. For each pair of the from $\langle\langle Z, j, k\rangle, w'\rangle$ in $\mathcal{S}$ where the grammar G conatins the production $X \stackrel{w''}{\rightarrow} YZ$ add the pair $\langle\langle X, i, k\rangle, w + w' + w''\rangle$ to $\mathcal{Q}$.
9. For each pair of the from $\langle\langle Z, k, j\rangle, w'\rangle$ in $\mathcal{S}$ where the grammar G conatins the production $X \stackrel{w''}{\rightarrow} ZY$ add the pair $\langle\langle X, k, j\rangle, w + w' + w''\rangle$ to $\mathcal{Q}$.
10. If $\mathcal{S}$ contains an assignment to $S, 1, n+1$ then terminate with success (a parse has been found) else terminate with failure (there is no parse).

8 Problem

Suppose you are given a curve model consisting of a sequence of positions x_1, x_2, ... x_n each of which is a position on the plane (a pair of real numbers). This sequence of positions is a curve model. We are indterested in finding curves of this shape in a given image. Consider a sequence of points in the image y_1, ... y_n. The score of placing the model points at these image positions is given as follows.

$$\begin{aligned}\text{cost} &= \sum_{i=1}^{n-2} \text{Shape-Cost}(x_i, x_{i+1}, x_{i+2}, y_i, y_{i+1}, y_{i+2}) \\ &\quad + \sum_{i=1}^{n-1} \text{Data-Cost}(y_i, y_{i+1})\end{aligned}$$

Give an algorithm for computing the cost of the least cost sequence y_1, ..., y_n. You should first give inference rules for computing, for a given i and image positions y_i and y_{i+1}, the cost $\text{cost}(i, y_i, y_{i+1})$, of the least cost sequence y_1, ..., y_i, y_{i+1}. You should be able to derive $\text{cost}(i+1, y_{i+1}, y_{i+2})$ given all values for $\text{cost}(i, y_i, y_{i+1})$. Give the Dijstra lightest derivation algorithm for this problem.