Itzel Orozco
June 28, 2024
CS 460

Finding PC Components Using A* Search Algorithm

Scenario: Someone wants to build a PC with a budget in mind but doesn't know how or where to get their research.
Code: Helps user make the decision by inputting their budget and getting a list of components as the result.
Algorithm: The A* Search Algorithm helps explore different combinations of components and find the best options that meet and exceed in both performance and budget.

My algorithm:

```
(1)def a_star_search(components, target_performance, max_budget):

(2) priority_queue = [(0, 0, {})]

(3) while priority_queue:

        current_cost, current_performance, selected_components =
heapq.heappop(priority_queue)

(4)     if current_performance >= target_performance and current_cost <=
max_budget:
            return selected_components

(5)     for category, options in components.items():
            for component in options:
                new_cost = current_cost + component['price']
                new_performance = current_performance +
component['performance']
(6)             if new_cost <= max_budget:
                    new_state = selected_components.copy()
                    new_state[category] = component
                    heapq.heappush(priority_queue, (new_cost + heuristic,
new_performance, new_state))

(7) return None
```

(1) We declare 3 parameters for the components, which comes from a section of different categories of components which also display the price and performance rate. For target performance, this is a parameter for the desired total performance that the selected components should achieve. Then lastly, budget is a parameter in which there is a limit and finds a way to fit as much in without passing the given budget.

(2) Initialized with a tuple: initial cost, total performance, and selected components.

(3) The function continues as long as there are elements in the priority queue. We extract current cost, performance, and components from the heap.

(4) Checking if current performance meets or exceeds target performance and if current cost is within budget. We then return the selected components.

(5) We iterate through each category of components. This calculates the new cost by adding the price of the component to the current cost. Then we calculate new performance by adding the performance of the component to current.

(6) Create an if statement to check if still within budget. We copy selected components into a new state and update it with the component that is selected for the current category. Calculate a heuristic value and push the new state into priority queue with the updated cost, performance and ‘new state’ aka the selected components.

(7) We return None if there are no valid combinations of components.