```
def uniformCostSearch(problem: SearchProblem):
    bucket = util.PriorityQueue()
    bucket.push((problem.getStartState(), []), 0)
    visited = []
    while not bucket.isEmpty():
        state_actions = bucket.pop()
        currentState = state_actions[0]
        acts = state_actions[1]
        visited.append(currentState)
        if problem.isGoalState(currentState):
            """
            The Total cost of path is calculated here on the basis of the actions
taken to reach the goal state.
            Cost of West and East is 1 and Cost of South and North is 2.
            """
            cost = 0
            for act in acts:
                if act == 'West' or act == 'East':
                    print(cost, " + 1 =", cost + 1, " for ", act)
                    cost += 1
                elif act == 'South' or act == 'North':
                    print(cost, " + 2 =", cost + 2, " for ", act)
                    cost += 2
            print("\nPath Found with Cost :",cost,"For UniformCostSearch\n")
            return acts
        for state, actions, cost in problem.getSuccessors(currentState):
            if state not in visited:
                c_actions = acts + [actions]
                newState = (state, c_actions)
                """
                Priority: West & East = 2, South & North = 1
                The priority is set on the basis of the actions taken to reach the
goal state.
                Priority of West and East is higher because it has less cost & vice
versa for North and South.
                Priority Queue is used So, it will pop the element with the highest
priority first.
                And as priority of West and East is higher and its cost is lower,
it will be popped first, and
                the path will be found with low cost.
                """
                priority = 0
                if actions == 'West' or actions == 'East':
                    priority = 2
                elif actions == 'South' or actions == 'North':
                    priority = 1
                bucket.push(newState, priority)
```