Objective
The purpose of the following handout is to walk you step by step through the execution of the FF planning algorithm, on a simple example. The FF algorithm is presented in the paper:


Problem
You are given operators for the door domain (move between rooms A and B, separated by a door), with the Polish operator added.

Door-2 Domain Operators

Move (x,y)
Pre: in(x)
Add: in(y)
Del: in(x)

Open
Pre: closed
Add: opened
Del: closed

Close
Pre: opened
Add: closed
Del: opened

Polish
Pre: opened
Add: polished
Del: 

Consider the problem of generating a plan that moves from the initial state:

In(A)
Closed
To the goal state:

| In(B) | Closed | Polished |

In the following we develop the tree of nodes visited in the state space search by the FF algorithm. We label each node with its relaxed distance estimate. For each node we demonstrate how the relaxed distance estimate is computed, including the plan graph generated from the node to the goal state, the relaxed plan that is extracted, and the resulting heuristic cost of that node.

FF starts with the initial state as the root of the tree:

**Node 1**

| In(A) | Closed |

FF then computes the heuristic value of Node 1 by generating a relaxed plan graph, from Node 1 to the goal:

Each operator is relaxed by simply eliminating its delete list. The relaxed plan graph is similar to that produced by GraphPlan, except that it does not contain any mutual exclusion relations. The relaxed plan graph for Node 1 consists of three fact layers and two action layers. The three goal propositions appear in the third fact layer.

Note that, for simplicity, in our drawing we have drawn each action once, in the action layer in which it first appears. That action should also appear in all subsequent layers. However, its consequences are also achieved through null operations (noops). For example, in the graph above, FF or GraphPlan would include an arc from Opened in layer 2 to Closed in layer 3, representing the Close action. When generating the relaxed plan, FF never selects these subsequent actions, since it always prefers noops to actions. Hence, they serve no useful role.
To generate the relaxed plan, FF starts with the three goal propositions:

\text{In(B), Closed, Polished.}

For each goal it selects a noop or action from action layer 2 that achieves it, preferring no-ops over actions. The selected actions/noops are highlighted in bold. \text{In(B)} is achieved by action \text{Move(A,B)}, \text{Closed} is carried forward by a noop, and \text{Polished} is achieved using the Polish action. The corresponding subgoals for fact layer 2 are collected, and consist of:

\text{In(A), Closed, Open}

Next, actions and noops from action layer 1 are selected to achieve these subgoals. \text{In(A)} and \text{Closed} are achieved by noops, while \text{Open} is achieved with the \text{Close} action. The corresponding subgoals are all propositions in the initial state.

The heuristic value of Node 1 is the number of actions in the relaxed plan. The complete relaxed plan, shown in bold, consists of three actions -- \text{Open}, \text{Move(A,B)} and \text{Polish} -- hence, the heuristic value of Node 1 is 3.

FF expands Node 1 by considering the application of each action in the first layer of the relaxed plan. Note that FF only uses an action in this first layer if the effects of the action are used by an action in layer 2. FF stops evaluating actions as soon as it finds an action whose target state has a better heuristic value. If the heuristic value does not improve, it performs a breadth first search.

For Node 1, only one action, \text{Closed}, appears in the first layer of the relaxed plan, hence it is the only action expanded. The expanded graph is shown below, with Node 2 added:

Next we generate the relaxed plan graph and plan for Node 2, as shown below:
The plan graph has one action layer, separating the current state and goal state fact layers. The relaxed plan consists of three parallel actions: **Move(A,B)**, **Close**, and **Polish**, resulting in a cost estimate of 3. The node value is the same as for Node 1, indicating a plateau. Hence, actions are explored in breadth-first order until the first action is found with decreased heuristic cost. In this example, we evaluate each of the three concurrent actions of Node 2’s plan, working in a top-down order, as they are written in the relaxed plan. However, no specific order for expanding the breadth-first search is required.

First we expand the action **Move(A,B)**:

Next we calculate the relaxed plan graph estimate for Node 3:
Once again the plan graph has a single action layer. The relaxed plan only contains two actions: \textbf{Close} and \textbf{Polish}, hence the cost of Node 3 is 2. This estimate is strictly better than that of Nodes 1 and 2; hence Node 3 moves FF off the plateau, and is chosen for further expansion. FF has two actions available to expand Node 3, either \textbf{Close} or \textbf{Polish}. First, choosing \textbf{Close}, Node 4 is added to the search tree:

The relaxed plan graph and plan for Node 4 is as follows:
Note that this plan graph has two action layers, one to open the door, and the second to act based on the open door (also note that the plan graph includes move operations, but are not indicated here for simplicity). The relaxed plan consists of Open, followed by Polish, with a heuristic cost of 2. This does not represent an improvement over Node 3, hence we try expanding Node 3 using action **Polish**, producing Node 5:

The relaxed plan graph for Node 5 is:
It has a single action, **Close**, hence the estimated cost for Node 5 is 1. This is strictly better than the cost at Node 3, so FF continues its expansion from Node 5. One action is available for expansion, **Close**, resulting in Node 6:
FF finds that all goal propositions are achieved at Node 6, hence the plan graph contains a single fact layer, and has an estimated cost of 0:

The completed plan is the action sequence from Node 1 to 6, consisting of **Open**, **Move(A,B)**, **Polish**, **Close**: