Problem 1.

(a)

```
automaton TokenMutualExclusion(N: Nat), where N > 1
  type ID: enumeration [0,...,N-1]
  type Val: enumeration [0,1,2,3]
  actions
      update(i:ID)
  variables
      s: [ID -> Val]
        initially (s[0] = 1 \/ s[0] = 3) /\ (s[N-1] = 0 \/ s[N
           -1] = 2) /\ (forall i: ID, 1 <= i <= N-2 -> s[i] =
           0)
   transitions

      update(i:ID)
        pre (i = 0 /\ (s[1] = (s[0] + 1) % 4))
             \/ (i = N-1 /\ (s[N-2] = (s[N-1] + 1) % 4))
        eff s[i] := (s[i] + 2) % 4

      update(i:ID)
        pre (0 < i < N-1) /\ ((s[i-1] = (s[i] + 1) % 4) \/ (s[i
           +1] = (s[i] + 1) % 4))
        eff s[i] := (s[i] + 1) % 4
```

(b)

```
initial state : [1,1,1,1,0]
step0 : [1,1,1,1,**0**] process 4 has token
step1 : [1,1,1,**1**,2] process 3 has token
step2 : [1,1,**1**,2,2] process 2 has token
step3 : [1,**1**,2,2,2] process 1 has token
step4 : [**1**,2,2,2,2] process 0 has token
step5 : [3,**2**,2,2,2] process 1 has token
step6 : [3,3,**2**,2,2] process 2 has token
step7 : [3,3,3,**2**,2] process 3 has token
step8 : [3,3,3,3,**2**] process 4 has token
step9 : [3,3,3,**3**,0] process 3 has token
...
```

(c)

```
initial state: [1,1,0,1,1,0]
step0 : [1,1,**0**,1,1,**0**] process 2 and 5 have token
step1 : [1,1,1,1,**1**,2] process 4 has token
```

```
step2 : [1,1,1,**1**,2,2] process 3 has token
step3 : [1,1,**1**,2,2,2] process 2 has token
step4 : [1,**1**,2,2,2,2] process 1 has token
step5 : [**1**,2,2,2,2,2] process 0 has token
step6 : [3,**2**,2,2,2,2] process 1 has token
...

(d)
We prove that for all k, at most one process holds the token at step k.

Base Case:
    The initial state contains exactly one process that holds the token,
    since the system is initialized correctly, the system has a single token.
    Thus, base case satisfies hypothesis

Inductive Step:
    for arbitrary positive integer k,
    assume at step k the invariant holds, meaning at most one process has the
    token
    By the update rule, only the process holding the token can update.
        When a process updates, it loses the token because its state changes,
        The token moves to a neighbor, but it does not duplicate.
    Since only one process i updates at step k+1, the token moves to at most
    one new process.
    Thus "The system has a single token" holds true for the k+1 step.

Conclusion:
    Thus we proof by induction, for all k, at most one process holds the
    token at step k.
```

Problem 2.

(1)

1. If feature A is selected, then either feature B or feature C must also be selected:

$$\neg A \vee B \vee C$$

2. Feature C and feature D cannot be selected together:

$$\neg C \vee \neg D$$

3. If feature E is selected, then feature F must not be selected:

$$\neg E \vee \neg F$$

4. At least one of features B, C, or D must be selected:

$$B \vee C \vee D$$

5. If feature F is selected, then either feature A or feature E must also be selected:

$$\neg F \vee A \vee E$$

6. At most 4 features can be selected:

$$A + B + C + D + E + F \leq 4$$

(2)

1. Since one of B,C,D must be selected, we arbitrarily choose B

$$Assign B = 1$$

2. Simplify $\neg A \vee B \vee C$, since B=1, this clause is always true and removed

$$Assign B = 1$$

3. Choose E = 1 arbitrarily, since $\neg F \vee A \vee E$, assign F=0

$$Assign B = 1, E = 1, F = 0$$

4. Choose C = 1 arbitrarily, since $\neg C \vee \neg D$, assign D=0;

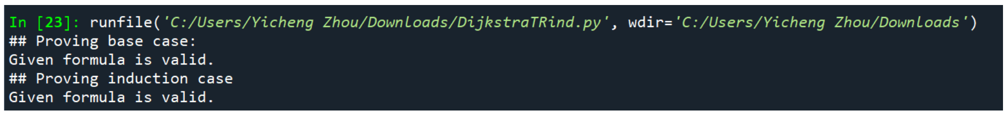
$$Assign B = 1, C = 1, D = 0, E = 1, F = 0$$

5. Since no constraint calls for A=1, arbitrarily choose A=0;

$$Assign A = 0, B = 1, C = 1, D = 0, E = 0, F = 0$$

6. check that all constraints are satisfied and A+B+C+D+E+F = 3 < 4

Problem 3.



```
In [23]: runfile('C:/Users/Yicheng Zhou/Downloads/DijkstraTRind.py', wdir='C:/Users/Yicheng Zhou/Downloads')
## Proving base case:
Given formula is valid.
## Proving induction case
Given formula is valid.
```

Figure 1: Terminal Output

Problem 4.

A. Target System: Our target system is a pipelined digital circuit implemented with RTL. We approach its timing verification statistically by modeling the system as a directed graph, where each node represents a pipeline register and each edge represents a timing path whose delay is described by a probability distribution. This approach accounts for process variations, environmental factors, and other uncertainties. Xilinx Vivado is used to generate the timing reports that form the basis of our statistical model [1]

   A.1. Model Availability: We Could developed simulation code that extracts a model from RTL descriptions and timing reports produced by Xilinx Vivado. model for analysis.

   A.2. Our model is implemented as simulation code.

B. Target Requirement: The requirement for our project is to verify that the pipelined design meets its timing constraints with a very low probability of failure.

   B.1. Expression as an Invariant/Temporal Property: The timing requirement is formalized as a probabilistic invariant stating that for every clock cycle, the probability of any timing path exceeding its allotted slack must be below a predetermined threshold.

   B.2. Primary Sources of Uncertainty: The main uncertainties in the design arise from process variations during manufacturing, environmental influences such as temperature fluctuations and voltage noise, and the inherent approximations in our statistical delay models.

   B.3. Verification Approach: Instead of using traditional worst-case analysis, our framework employs statistical reasoning. Through techniques like Monte Carlo simulation, we estimate the likelihood of timing violations, which allows us to pursue more aggressive performance margins while still managing risk effectively.

C. Related Work and Innovation: Research in statistical timing analysis and probabilistic verification has paved the way for our project. Notable papers in this field include:

   1. "unified framework for statistical timing analysis with coupling and multiple input switching"[2]
   2. "Low power design flow with static and statistical timing analysis" [3]
   3. "Speeding up Monte-Carlo Simulation for Statistical Timing Analysis of Digital Integrated Circuits"[4]
   4. "Statistical timing analysis of combinational logic circuits" [5]
   5. "On Timing Model Extraction and Hierarchical Statistical Timing Analysis" [6]

References

[1] A. Xilinx, “Vivado design analysis: Timing analysis,” https://docs.amd.com/r/en-US/ug906-vivado-design-analysis/Timing-Analysis, AMD/Xilinx, n.d., accessed: 2025-02-17.

[2] J. K. Liou, K. T. T. Cheng, and S. R. Nassif, “A unified framework for statistical timing analysis with coupling and multiple input switching,” in *Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2005, pp. 134–140. [Online]. Available: https://ieeexplore.ieee.org/document/1560179

[3] K. C. Kuo and H. T. Ko, “Low power design flow with static and statistical timing analysis,” in *2012 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*. IEEE, 2012, pp. 798–801. [Online]. Available: https://ieeexplore.ieee.org/document/6473600

[4] S. R. Naidu, “Speeding up monte-carlo simulation for statistical timing analysis of digital integrated circuits,” in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID’07)*. IEEE, 2007, pp. 265–270. [Online]. Available: https://ieeexplore.ieee.org/document/4076190

[5] H. F. Jyu, S. Malik, S. Devadas, and K. W. Keutzer, “Statistical timing analysis of combinational logic circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, pp. 126–137, June 1993.

[6] B. Li, N. Chen, Y. Xu, and U. Schlichtmann, “On timing model extraction and hierarchical statistical timing analysis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 3, pp. 367–380, March 2013.