

## Lab 3: Performance modelling

Students: August Heltne-Karlson and Erik Turøy Midtun

Task 1: Analytic solution of small birth-death Markov chain using Mathematica (Warm up) [0%]

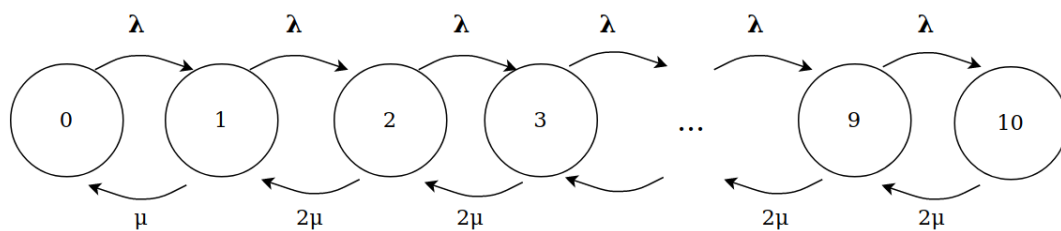
Task 2: Runways [25%]

Let the *system state* be the *number of airplanes* in landing and in the airspace over the airport.

NOTE: Takeoffs are not included in this part of the assignment.

The airspace has a limited capacity. Arriving airplanes will be *redirected* if both runways are in use and the airspace is full.

**2.1. Draw the Markov chain model for two runways ( $n=2$ ) and  $N=10$ . What kind of queue model is this in Kendall's notation?**



M/M/2/10/FIFO queue model

**2.2. Set up the equations and solve them to determine the steady state probabilities for  $n=2$  and  $N=10$ .**

```
In[*]:= vars = p# & /@ Range[0, 10]
```

```
Out[*]:= {p0, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10}
```

```

In[ ]:= eqns = {
  λ p0 == μ p1,
  λ p1 == 2 μ p2,
  λ p2 == 2 μ p3,
  λ p3 == 2 μ p4,
  λ p4 == 2 μ p5,
  λ p5 == 2 μ p6,
  λ p6 == 2 μ p7,
  λ p7 == 2 μ p8,
  λ p8 == 2 μ p9,
  λ p9 == 2 μ p10,
  vars . Table[1, {11}] == 1
};

sol = Solve[eqns, vars] // Flatten

Out[ ]:= {p0 → (512 μ10) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 +
  16 λ6 μ4 + 32 λ5 μ5 + 64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p1 → (512 λ μ9) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p2 → (256 λ2 μ8) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p3 → (128 λ3 μ7) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p4 → (64 λ4 μ6) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p5 → (32 λ5 μ5) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p6 → (16 λ6 μ4) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p7 → (8 λ7 μ3) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p8 → (4 λ8 μ2) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p9 → (2 λ9 μ) / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 +
  64 λ4 μ6 + 128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10),
  p10 → λ10 / (λ10 + 2 λ9 μ + 4 λ8 μ2 + 8 λ7 μ3 + 16 λ6 μ4 + 32 λ5 μ5 + 64 λ4 μ6 +
  128 λ3 μ7 + 256 λ2 μ8 + 512 λ μ9 + 512 μ10) }

```

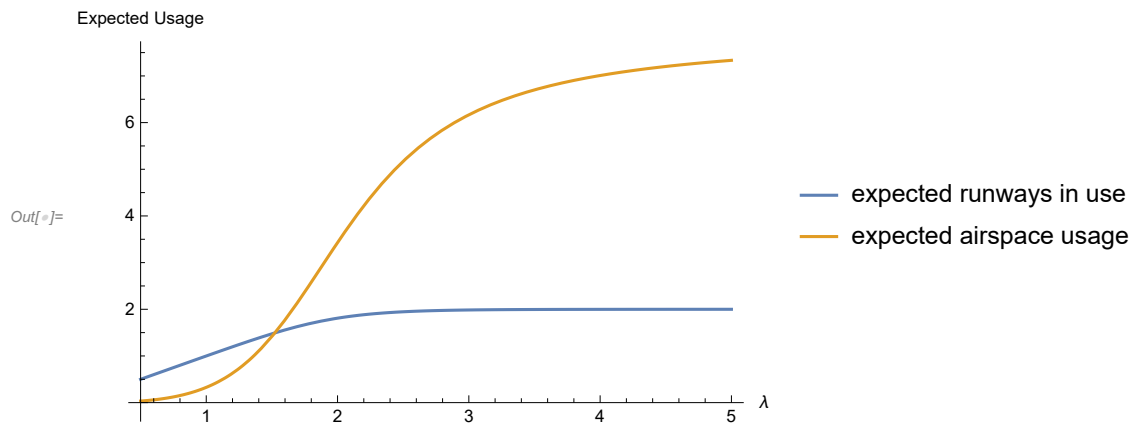
**2.3. Solve the equations and obtain the expected number of runways in use and the expected number of planes in the airspace, as a function of  $\lambda$  {plot for  $\lambda \in [0.5, 5]$ } with  $\mu=1$ .**

**Explain how and why the utilisation flattens out, and and argue for what you regard as an acceptable arrival rate  $\lambda$ .**

```

In[ ]:= para = {n → 2, N → 10};
(*expRunways=Σ{i,0,N}Min[i,n]*pi/.para*)
expRunwaysMask = {0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2};
expAirspaceMask = {0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8};
Plot[{Legended[AGen[λ, 1, expRunwaysMask], "expected runways in use"],
      Legended[AGen[λ, 1, expAirspaceMask], "expected airspace usage"]},
      {λ, 0.5, 5}, AxesLabel → {λ, "Expected Usage"}]

```



The utilisation of the runways and the airspace flatten out because they fill up. The runways have a capacity of maximum 2 planes handled at the same time because each runway can handle one plane at a time. The expected value of runways flattens out when it reaches two because the moment a plane leaves the runway, a new plane from the airspace enters the runway. Based on the graph above, a lambda value between 2 and 3 would give acceptable waiting times for the runway without wasting resources. The most important thing to note is that a lambda value of around 5 will result in a full airspace and force the airport to reject planes. This should be avoided.

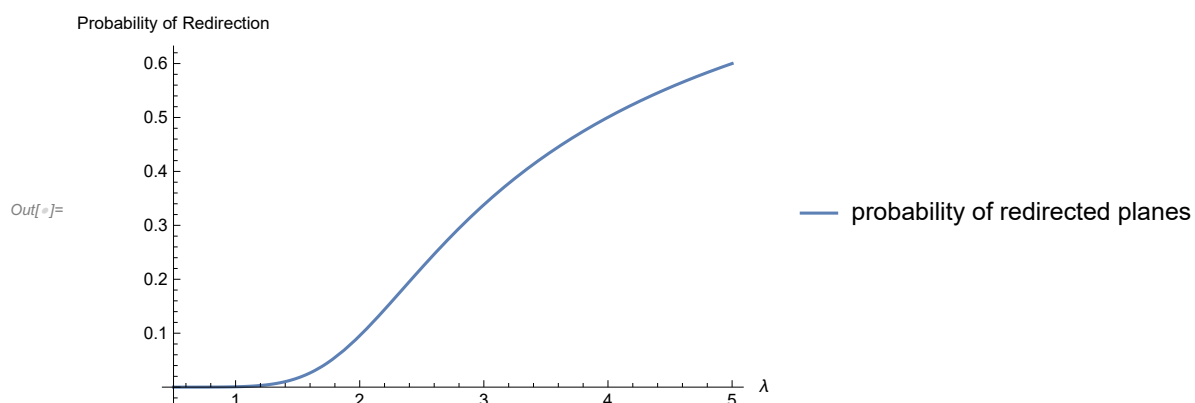
#### 2.4. Plot the probability of redirected airplanes ( $n=2$ , $N=10$ ) as a function of $\lambda \in [0.5, 5]$ (with $\mu=1$ ).

Argue for what you regard as an acceptable arrival rate  $\lambda$ .

```

In[ ]:= lossMask = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1};
Plot[{Legended[AGen[λ, 1, lossMask], "probability of redirected planes"]},
      {λ, 0.5, 5}, AxesLabel → {λ, "Probability of Redirection"}]

```



Given that rejecting an airplane is really bad for the passengers, the probability of being redirected should be as close to zero as possible. Therefore we suggest that an arrival rate between 1 and 1.25 should be chosen. While this comes at a cost to the total amount of planes that arrives per day, most passengers value dependability highly and would rather have fewer flights to choose between

instead of being forced to land in another city.

## 2.5. Make a function in Mathematica which implements Erlang's C formula ( $E_{2,n}(A)$ in Eq.(6.95) in the textbook).

Check that it returns  $E_{2,2}(A) = \frac{A^2}{2+A}$  ( $n=2$  and where  $A = \frac{\lambda}{\mu}$ ).

The probability of waiting in a M/M/n -queue is given by Erlang's C formula.

```
In[ ]:= Er1C[A_, n_] :=
  ((A^n) / n!) * n / (n - A) / (Sum[(A^i) / i! + ((A^n) / n!) * n / (n - A)]
```

```
para3 = {A -> λ / μ};
erlangCTest = Er1C[A, 2] // Simplify
```

```
Out[ ]:=
  A^2
  ---
  2 + A
```

## 2.6. Use the appropriate formulas from Chapter 6 to obtain the expected (sojourn) time $E[S]$ in the system when $n=2$ and $N$ is infinite.

From Eq.(6.100) in the book we get the following function:

```
In[ ]:= ExpSoujourn[A_, n_, μ_] := (1 / ((n - A) * μ)) * (Er1C[A, n] + n - A)
```

## 2.7. Plot the probability that both runways are free as a function of $\lambda \in [1,3]$ (with $\mu=1$ ) with $N=10$ and $N$ infinite. (Comment results).

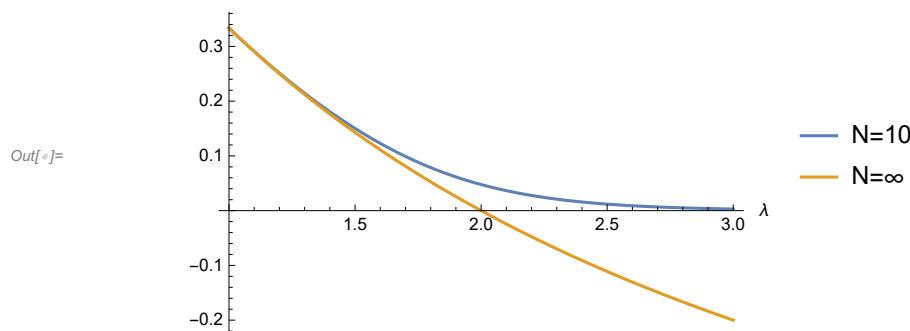
The probability that both runways are free is the same as  $p_0$ .

```
In[ ]:= SteadyStateForMMn[A_, n_, i_] :=
  If[i < n, ((A^i) / i!) / (Sum[(A^j) / j! + ((A^n) / n!) * (n / (n - A))]),
  ((A^i) / n! n^(i - n)) / (Sum[(A^j) / j! + ((A^n) / n!) * (n / (n - A))])]
```

```
p0mmnNinf = SteadyStateForMMn[A, 2, 0] /. {A -> λ / 1} // Simplify
emptyMaskN10 = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
Plot[{Legended[AGen[λ, 1, emptyMaskN10], "N=10"], Legended[p0mmnNinf, "N=∞"]},
  {λ, 1, 3}, AxesLabel -> {λ, "Probability that both runways are empty"}]
```

```
Out[ ]:=
  2 - λ
  ---
  2 + λ
```

Probability that both runways are empty

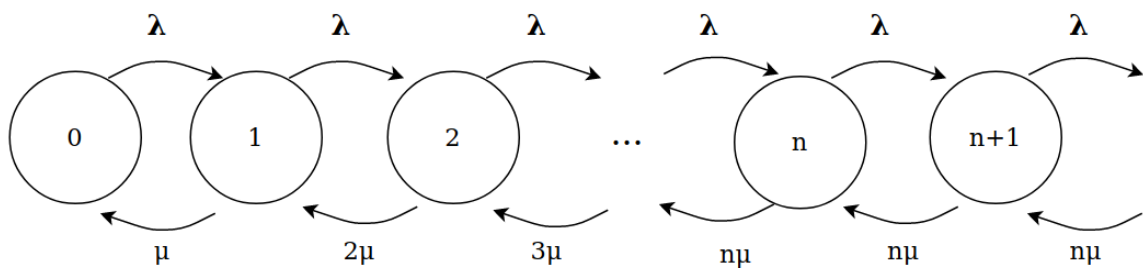


With a finite queue there's always the possibility that all planes in the airspace get to land before another plane arrives. This can be seen in the  $N = 10$  graph where the probability is very low, but non-zero. For  $N = \infty$ , the probability of both runways being free is 0 at  $\lambda = 2$  and becomes negative afterwards. This is caused by the system being unstable. For  $M/M/2$ , the following condition must be met for the system to be stable:  $2 > \lambda/\mu$ . With  $\mu = 1$  and  $\lambda \geq 2$ , this condition is not met and the queue will approach  $\infty$ .

### Task 3: Deicing station [20%]

The objective in this task is to check the effect of changing the number of deicing stations,  $n$ . Assume infinite queuing capacity.

**3.1. Draw the Markov chain model for  $n$  deicing stations. What kind of queue model is this in Kendall's notation?**



$M/M/n/\infty/FIFO$

**3.2. Set up the equations and solve them to provide the steady state probabilities for  $n=1$ .**

```
In[ ]:= vars = p# & /@ Range[0, 10];
```

```
eqns3an = {
  λ p0 == μ p1,
  λ p1 == Min[2, n] μ p2,
  λ p2 == Min[3, n] μ p3,
  λ p3 == Min[4, n] μ p4,
  λ p4 == Min[5, n] μ p5,
  λ p5 == Min[6, n] μ p6,
  λ p6 == Min[7, n] μ p7,
  λ p7 == Min[8, n] μ p8,
  λ p8 == Min[9, n] μ p9,
  λ p9 == Min[10, n] μ p10,
  vars . Table[1, {11}] == 1
};
```

```
sol1 = Solve[eqns3an /. {n → 1}, vars] // Flatten
```

$$\text{Out[ ]} = \left\{ \begin{aligned} p_0 &\rightarrow \frac{\mu^{10}}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_1 &\rightarrow \frac{\lambda \mu^9}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_2 &\rightarrow \frac{\lambda^2 \mu^8}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_3 &\rightarrow \frac{\lambda^3 \mu^7}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_4 &\rightarrow \frac{\lambda^4 \mu^6}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_5 &\rightarrow \frac{\lambda^5 \mu^5}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_6 &\rightarrow \frac{\lambda^6 \mu^4}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_7 &\rightarrow \frac{\lambda^7 \mu^3}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_8 &\rightarrow \frac{\lambda^8 \mu^2}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_9 &\rightarrow \frac{\lambda^9 \mu}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}}, \\ p_{10} &\rightarrow \frac{\lambda^{10}}{\lambda^{10} + \lambda^9 \mu + \lambda^8 \mu^2 + \lambda^7 \mu^3 + \lambda^6 \mu^4 + \lambda^5 \mu^5 + \lambda^4 \mu^6 + \lambda^3 \mu^7 + \lambda^2 \mu^8 + \lambda \mu^9 + \mu^{10}} \end{aligned} \right\}$$

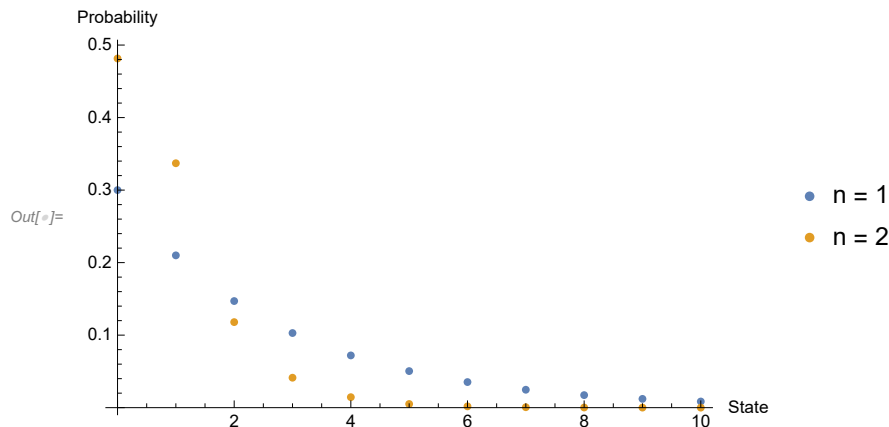
**3.3. Use the functions from Queueing Processes to obtain the steady state probabilities for  $n=1$  and  $n=2$ .**

**Plot (use ListPlot) the steady state probabilities of states 0-10 for  $n=1$  and  $n=2$ . ( $\lambda=0.07, \mu=0.1$ ). Compare and comment the difference between  $n=1$  and  $n=2$ .**

```

In[ ]:= P = {λ → 0.07, μ → 0.1};
(*QueueProperties[Q] //Simplify; *)
D1 = StationaryDistribution[QueueingProcess[λ, μ, 1, Infinity]];
D2 = StationaryDistribution[QueueingProcess[λ, μ, 2, Infinity]];
(* Numerical probabilities of all states *)
steadyStateN1 = Probability[n == #, n ≈ D1] & /@ Range[0, 10] /. P;
steadyStateN2 = Probability[n == #, n ≈ D2] & /@ Range[0, 10] /. P;
ListPlot[{Legended[steadyStateN1, "n = 1"], Legended[steadyStateN2, "n = 2"]},
  DataRange → {0, 10}, AxesLabel → {"State", "Probability"}]

```



We can see that the probability having a large queue decreases faster for  $n = 2$  than for  $n = 1$ . This is due to the fact that for  $n = 2$ , airplanes leave the system with a rate of  $2\mu$  compared to  $n = 1$  where the rate is  $\mu$ . As such  $n = 2$  has a higher probability of being in a state with few airplanes in the queue and a lower probability of being in a state with a lot of airplanes in the queue.

### 3.4. Use the functions from Queueing Processes to obtain a symbolic expression the time at deicing, $E[S]$ for $n$ .

```

In[ ]:= Q = QueueingProcess[λ, μ, n, Infinity];
QueueProperties[Q] // Simplify;
QueueProperties[Q, "MeanSystemTime"] // Simplify

```

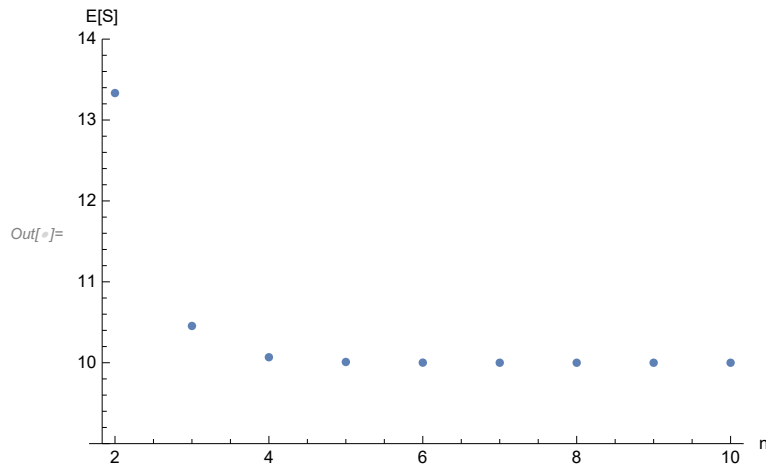
$$\text{Out[ ]} = \frac{1}{\mu} + \frac{n \left( \frac{\lambda}{\mu} \right)^n \mu \text{Gamma}[n]}{(-\lambda + n \mu) \left( n \left( \frac{\lambda}{\mu} \right)^n \mu \text{Gamma}[n] - e^{\lambda/\mu} (\lambda - n \mu) n! \text{Gamma}\left[n, \frac{\lambda}{\mu}\right] \right)}$$

**3.5. Plot the  $E[S]$  for  $n=1,...,10$  (deicing stations), using the following parameters :  $\{\lambda=0.1, \mu=0.1\}$ . How does the  $E[S]$  change with  $n$ ? How many deicing stations would you recommend (argue)?**

```

In[ ]:= P = {λ → 0.1, μ → 0.1};
Q = QueueingProcess[λ, μ, n, Infinity];
queue = QueueProperties[Q, "MeanSystemTime"] /. P // Simplify;
probs1 = Table[{n, queue}, {n, 2, 10}];
ListPlot[{probs1}, DataRange → {2, 10},
  PlotRange → {9, 14}, AxesLabel → {"n", "E[S]"}]

```



If the system only has a single deicing truck the condition  $1 > A$  is not met and the system is therefore not stable. The result for a single truck is therefore omitted from the graph. Based on the graph above, one would either pick three or four deicing trucks as those yield the highest return on investment. Five deicing trucks are barely better than four trucks and the expected time for two trucks is much higher than the expected time for three trucks.

#### Task 4: Turnaround at gate [15%]

After landing, the airplane is taxiing to the gate where unloading and reloading take place, before the plane leaves the gate and is ready for takeoff. The the expected turnaround time is  $1/\mu_t$ . There is always a gate available for an arriving airplane.

**4.1. Suggest a queueing model for the gates (you may use Kendall's notation)**

M/M/∞/∞/FIFO

**4.2. Obtain (on paper if you like) a symbolic expression for the steady state probabilities.**

**What is this probability distribution called?**

From the book we know that the steady state probabilities in a M/M/∞ system are:

**Steady state probabilities in an M/M/∞ system**

$$p_i = e^{-A} \frac{A^i}{i!}, \forall i \in \mathbb{N} \quad (6.23)$$

This is a Poisson distribution with A as the parameter with  $A = \lambda/\mu$ .

**4.3. Use the "QueueingProcess" to obtain the same steady state probabilities.**



```

In[ ]:= Q = QueueingProcess[λ, μ, Infinity, Infinity];
QueueProperties[Q] // Simplify;
D = StationaryDistribution[Q];
Probability[n == i, n ~ D]

```

$$\text{Out[ ]} = \begin{cases} \frac{e^{-\frac{\lambda}{\mu}} \left(\frac{\lambda}{\mu}\right)^i}{i!} & i == \text{Floor}[i] \ \&\& \ i \geq 0 \\ 0 & \text{True} \end{cases}$$

We can see that this expression is the same as the one we found in 4.2 with  $A$  substituted by  $\lambda/\mu$ .

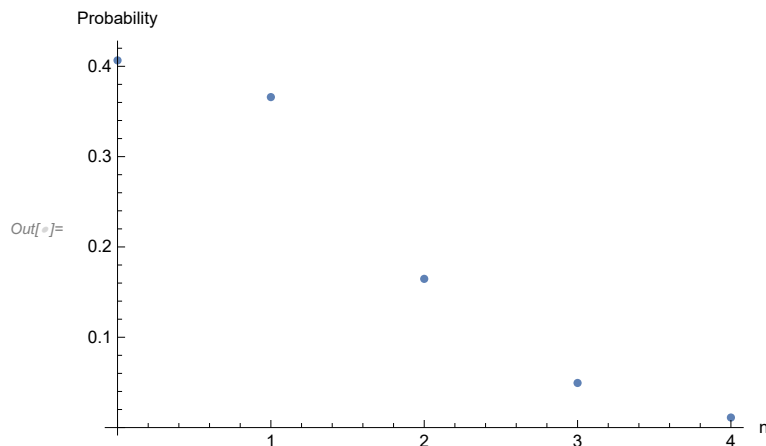
#### 4.4. Plot the probability that $n=0, 1, 2, 3, 4$ gates are occupied.

(para = {λ → 0.9, μ<sub>t</sub> → 1})

```

In[ ]:= P = {λ → 0.9, μt → 1};
Q = QueueingProcess[λ, μt, Infinity, Infinity];
D = StationaryDistribution[Q];
px := PDF[D, x];
probs = p# & /@ Range[0, 4] /. P;
ListPlot[{probs}, DataRange → {0, 4}, AxesLabel → {"n", "Probability"}]

```



#### 4.5. Obtain the $E[W]$ (time in the system). Compare with the expected service time $\left(\frac{1}{\mu_t}\right)$ and comment on the result.

```

In[ ]:= QueueProperties[Q, "MeanSystemTime"] // Simplify

```

$$\text{Out[ ]} = \frac{1}{\mu_t}$$

As the system has infinite capacity no time will be spent waiting for the service and there is no queueing to leave the system. As such  $E[W]$  is equal to  $E[S]$ .

### Task 5: The airport in cold weather [40%]

In this task it is expected that you solve the problem without help from the assistants (only to clarify any ambiguities and typos) - you're on your own...

In our system (the airport), an airplane needs

- access to a runway twice (landing and takeoff),
- a gate for turnaround of passengers and goods,

- and in case of cold weather it also needs a deicing station.

In this task you should assume that there is cold weather (but no snow). The runways are shared between landing and takeoff and in this analytic model we assume no priority between the two.

Model the total “life cycle” of an airplane a network of queues using the Jackson’s theorem [[Mathematica has a “QueueingNetworkProcess” which is useful]].

### 5.1. Explain how to apply the Jackson’s theorem and list the assumptions

The main application of the Jackson’s theorem is that it allows us to create a network of nodes and still study the individual nodes independently after we have determined  $A_k$  for all the nodes.

Conditions needed for Jackson’s Theorem:

1. Customers arrive from outside the system at node  $k$  according to a Poisson process with intensity  $\lambda_k$ .
2. The number of servers in node  $k$  is  $n_k \geq 1$  where  $k = 1, \dots, K$ . The service time in node  $k$  is negative exponentially distributed with parameter  $\mu_k$ .

These conditions are both fulfilled as all the nodes are M/M/n with  $n \geq 1$ .

3. Each node has an infinite number of queueing positions.

We can fulfill this condition by modelling all the queues as M/M/n/ $\infty$ .

4. After a service completion at node  $k$  a customer is either instantaneously routed to and received at node  $j$  with probability  $p_{kj}$ , or leaves instantaneously the system with probability  $p_{k0} = 1 - \sum_{j=1}^K p_{kj}$ .

This condition is fulfilled as all customers leaving the runway the first time are instantly sent to the gate. After finishing at the gate the customer is instantly sent to the deicing station and after finishing at the deicing station the plane returns to the runway. When leaving the runway the second time, the customer leaves the system instantly.

### 5.2. What are the “customers” and “nodes” in the system?

The nodes are the runways, the deicing stations and the gates.

The customers are the planes as they are requesting service from the nodes.

### 5.3. Suggest a queue model type (M/M/?) for each node (the nodes might need different kinds of queue model)?

Runways: M/M/2/ $\infty$ /FIFO

We use the same variables for runways as in Task 2 with the queue capacity increased to  $\infty$  due to condition 3.

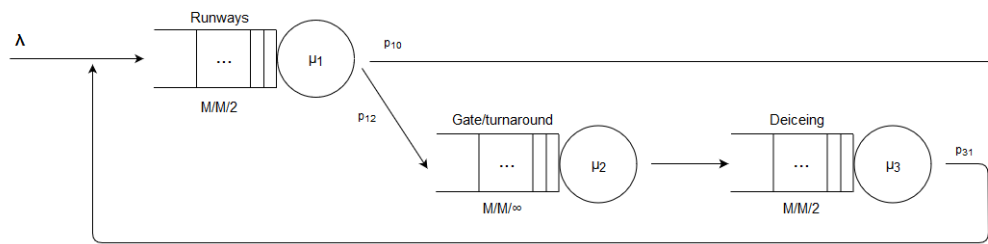
Deicing Stations: M/M/2/ $\infty$ /FIFO

Due to us observing that the deicing stations required  $n > 1$  to be stable, we use two deicing stations.

Gates: M/M/ $\infty$ / $\infty$ /FIFO

Gates keep the same variables from Task 4.

#### 5.4. Draw the model (show the network of queues and indicate what type of queue you have in each node)



#### 5.5. Define the total arrival intensity for each of the queues (nodes) in the network of queues

```
In[ ]:= vars = {Λ₁, Λ₂, Λ₃};
eqns = {
  Λ₁ == λ₁ + Λ₃,
  Λ₂ == λ₂ + p₁₂ Λ₁,
  Λ₃ == λ₃ + Λ₂
};
sol = Solve[eqns, vars];
para5 = {λ₁ → 1, λ₂ → 0, λ₃ → 0, p₁₂ → 0.5, μ → 0};
nsol3 = sol /. para5
```

```
Out[ ]:= {{Λ₁ → 2., Λ₂ → 1., Λ₃ → 1.}}
```

The cell above is quite unstable due to other functions. The proper output is

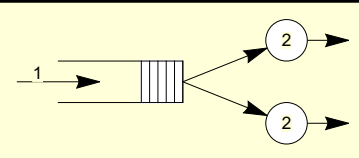
```
{Λ₁ → 2., Λ₂ → 1., Λ₃ → 1.}.
```

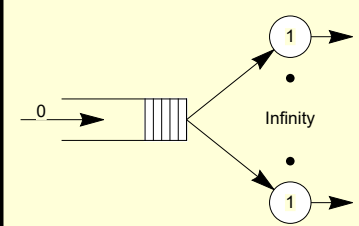
#### 5.6. Apply "QueueingNetworkProcess" to specify the network model

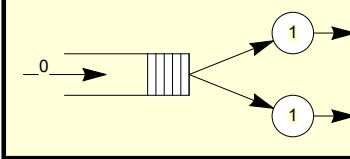
```
In[ ]:= γ = {1, 0, 0};
μ = {2, 1, 1};
r = {{0, 0.5, 0}, {0, 0, 1}, {1, 0, 0}};
c = {2, Infinity, 2};

N = QueueingNetworkProcess[γ, r, μ, c];
Table[QueueProperties[{N, i}], {i, 3}]
node1 = QueueingProcess[2, 2, 2];
node2 = QueueingProcess[1, 1, Infinity];
node3 = QueueingProcess[1, 1, 2];
```

Out[\*]= {

	
<b>Basic Properties</b>	
NetworkType	Open
NodeCount	3
SelectedNode	1
<b>Performance Measures</b>	
MeanSystemSize	1.33333
MeanSystemTime	0.666667
MeanQueueSize	0.333333
MeanQueueTime	0.166667

	
<b>Basic Properties</b>	
NetworkType	Open
NodeCount	3
SelectedNode	2
<b>Performance Measures</b>	
MeanSystemSize	1.
MeanSystemTime	1.
MeanQueueSize	0.
MeanQueueTime	0.

	
<b>Basic Properties</b>	
NetworkType	Open
NodeCount	3
SelectedNode	3
<b>Performance Measures</b>	
MeanSystemSize	1.33333
MeanSystemTime	1.33333
MeanQueueSize	0.333333
MeanQueueTime	0.333333

**5.7. What is the probability that both runways are idle?**

```

In[ ]:= network = QueueingNetworkProcess[γ, r, μ, c];
distribution = StationaryDistribution[network];
Total[Total[Table[PDF[distribution, {0, i, k}], {i, 0, 20}, {k, 0, 20}]]]
PDF[StationaryDistribution[node1], 0]

```

Out[ ]:= 0.333333

Out[ ]:=  $\frac{1}{3}$

As the network is a Jackson network, we can examine the nodes independently or a part of the system. As shown above, both of these methods will lead to the same answer. This property is also used for the following two questions.

### What is the probability that there are 4 airplanes?

```

In[ ]:= combos = Select[Tuples[Range[0, 4], 3], Total@# ≤ 4 &];
subs = Select[combos, Total[#] == 4 &];
probs = Table[PDF[distribution, sub], {sub, subs}];
Grid[{subs, probs}]
Total[probs]

```

{0, 0, 4}	{0, 1, 3}	{0, 2, 2}	{0, 3, 1}	{0, 4, 0}	{1, 0, 3}	{1, 1, 2}	{1, 2, 1}	{1, 3, 0}	{2, 0, 2}	{2, 1, 1}	{2, 2, 0}	{3, 0, 1}	{3, 1, 0}	{4, 0, 0}
0.05	0.10	0.10	0.06	0.01	0.20	0.20	0.06	0.10	0.20	0.10	0.10	0.10	0.10	0.05
10	21	21	81	70	21	43	43	81	21	43	21	21	21	10
94	89	89	25	31	89	77	77	25	89	77	89	89	89	94
4			8	5			8							4

Out[ ]:= 0.158393

We assume that the question is “What is the probability that there are 4 airplanes in the system at the same time?”. To solve this, we calculate the sum of the probability of each permutation of airplanes in the system that has a sum of 4. This cell has a tendency to give out unreadable output, but the total output is:

```

Out[54]= {0, 0, 4} {0, 1, 3} {0, 2, 2} {0, 3, 1} {0, 4, 0} {1, 0, 3} {1, 1, 2} {1, 2, 1} {1, 3, 0}
0.00510944 0.0102189 0.0102189 0.00681258 0.00170315 0.0102189 0.0204377 0.0204377 0.00681258

```

Out[55]= 0.158393

### What is the probability that one airplane is at the deicing station?

```

In[ ]:= distribution = StationaryDistribution[node3];
PDF[distribution, 1]

```

Out[ ]:=  $\frac{1}{3}$

## 5.8. Obtain an expression for the expected number of airplanes in at the airport

```

γ = {λ, 0, 0};
μ = {2, 1, 1};
r = {{0, 0.5, 0}, {0, 0, 1}, {1, 0, 0}};
c = {2, Infinity, 2};
N = QueueingNetworkProcess[γ, r, μ, c];
Total[Table[QueueProperties[{N, i}, "MeanSystemSize"], {i, 3}]] // Simplify

```

$$\text{Out}[*]= 3. \lambda + \frac{\lambda^3 \left( -\frac{1.}{1. \lambda^2 + e^{1. \lambda} (2. - 1. \lambda) \text{Gamma}[2, 1. \lambda]} - \frac{1.}{1. \lambda^2 + e^{1. \lambda} (2. - 1. \lambda) \text{Gamma}[2, 0. + 1. \lambda]} \right)}{-2. + 1. \lambda}$$

### 5.9. Obtain an expression for the expected time at the airport, $E[S_a]$ .

```

In[*]:= γ = {λ, 0, 0};
μ = {2, 1, 1};
r = {{0, 0.5, 0}, {0, 0, 1}, {1, 0, 0}};
c = {Nrun, Infinity, Nice};
N = QueueingNetworkProcess[γ, r, μ, c];
expectedTimeAtAirport =
  Total[Table[QueueProperties[{N, i}, "MeanSystemTime"], {i, 3}]] // Simplify

```

$$\text{Out}[*]= 2.5 + \frac{1. \lambda^{N_{\text{ice}}}}{N_{\text{ice}}! \left( \frac{e^{1. \lambda} \text{Gamma}[N_{\text{ice}}, 0. + 1. \lambda]}{\text{Gamma}[N_{\text{ice}}]} + \frac{1. \lambda^{N_{\text{ice}}}}{N_{\text{ice}}! - \frac{1. \lambda N_{\text{ice}}!}{N_{\text{ice}}}} \right) \left( 1 - \frac{1. \lambda}{N_{\text{ice}}} \right)^2 N_{\text{ice}}} + \frac{0.5 \lambda^{N_{\text{run}}}}{N_{\text{run}}! \left( \frac{e^{1. \lambda} \text{Gamma}[N_{\text{run}}, 0. + 1. \lambda]}{\text{Gamma}[N_{\text{run}}]} + \frac{1. \lambda^{N_{\text{run}}}}{N_{\text{run}}! - \frac{1. \lambda N_{\text{run}}!}{N_{\text{run}}}} \right) \left( 1 - \frac{1. \lambda}{N_{\text{run}}} \right)^2 N_{\text{run}}}$$

### 5.10. Plot the expected time at the airport for different $\lambda$ values.

Argue for what you regard as an acceptable arrival rate  $\lambda$  wrt. to total time at airport for an airplane.

In[ ]:=

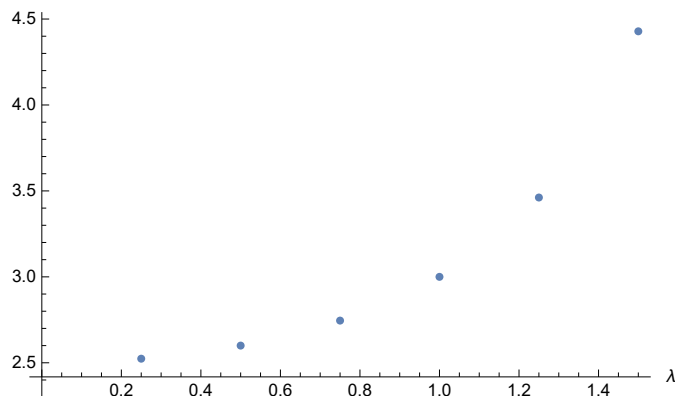
```

expectedTimeAtAirportWithParam = expectedTimeAtAirport /. {Nrun → 2, Nice → 2};
ListPlot[Table[expectedTimeAtAirportWithParam, {λ, {0.25, 0.5, 0.75, 1, 1.25, 1.5}}],
  DataRange → {0.25, 1.5}, AxesLabel → {λ, "Expected Time"}]
Grid[{{0.25, 0.5, 0.75, 1, 1.25, 1.5},
  Table[expectedTimeAtAirportWithParam, {λ, {0.25, 0.5, 0.75, 1, 1.25, 1.5}}]}]

```

Expected Time

Out[ ]:=



Out[ ]:=

```

0.25 0.5 0.75 1 1.25 1.5
2.52381 2.6 2.74545 3. 3.46154 4.42857

```

With the variables used for this network we can see that the expectedTimeAtAirport rises exponentially with  $\lambda$ . When  $\lambda$  reaches 2, the system will become unstable and the expected time will approach  $\infty$ .

In order to maximize performance we want to pick a  $\lambda$  such that  $\lambda$  is high and the expected time is low. This can be done by calculating and comparing values for  $\lambda/E[S]$ :

Lambda:	0.25	0.5	0.75	1	1.25	1.5
E[S]	2.52381	2.6	2.74545	3	3.46154	4.42857
Lambda/E[S]:	0.099	0.192	0.273	0.333	0.361	0.339

Based on this, picking  $\lambda = 1.25$  would be the best choice for optimizing performance. The truly optimal  $\lambda$  can be found by analyzing the function, but we regard this as out of scope and it will only result in minor gains.

**5.11 Investigate where the bottleneck is and discuss which changes can be made (in service rates, resource capacities) if the arrival intensity  $\lambda$  is fixed and the total time at the airport  $E[S_a]$  is regarded as too high.**

There are two ways we can reduce the total time at the airport:

1. Reduce time spent in queues by increasing capacity
2. Reduce time spent in service by reducing service time. This will also reduce time spent in queues, but is harder to do in a real life scenario.

```

In[ ]:=  $\gamma = \{\lambda, 0, 0\};$ 
 $\mu = \{2, 1, 1\};$ 
 $r = \{\{0, 0.5, 0\}, \{0, 0, 1\}, \{1, 0, 0\}\};$ 
 $c = \{N_{run}, \text{Infinity}, N_{ice}\};$ 
 $N = \text{QueueingNetworkProcess}[\gamma, r, \mu, c];$ 
expectedTimeAtAirport =
  Total[Table[QueueProperties[{N, i}, "MeanSystemTime"], {i, 3}]] // Simplify;
Print["Increasing the number of deicing stations"]
Table[expectedTimeAtAirport /. { $\lambda \rightarrow 1.25$ ,  $N_{run} \rightarrow 2$ ,  $N_{ice} \rightarrow i$ }, {i, 2, 10}]
Print["Increasing the number of runways"]
Table[expectedTimeAtAirport /. { $\lambda \rightarrow 1.25$ ,  $N_{run} \rightarrow i$ ,  $N_{ice} \rightarrow 2$ }, {i, 2, 10}]

Increasing the number of deicing stations
Out[ ]:= {3.46154, 2.90935, 2.83586, 2.8231, 2.82092, 2.82057, 2.82052, 2.82051, 2.82051}

Increasing the number of runways
Out[ ]:= {3.46154, 3.18545, 3.1487, 3.14232, 3.14123, 3.14105, 3.14103, 3.14103, 3.14103}

```

### Increasing Capacity:

For the number of deicing stations, we get the same result as in Task 3.5 with the increase flattening out after 4 deicing stations.

Increasing the amount of runways has a lesser effect compared to increasing the number of deicing stations, but going up to three runways is still a decent way to save time compared to two runways. Increasing capacity is irrelevant for gates as we assume infinite capacity.

```

In[ ]:=  $\gamma = \{\lambda, 0, 0\};$ 
 $\mu = \{u_{runway}, u_{gate}, u_{deicing}\};$ 
 $r = \{\{0, 0.5, 0\}, \{0, 0, 1\}, \{1, 0, 0\}\};$ 
 $c = \{2, \text{Infinity}, 2\};$ 
 $N = \text{QueueingNetworkProcess}[\gamma, r, \mu, c];$ 
expectedTimeAtAirport =
  Total[Table[QueueProperties[{N, i}, "MeanSystemTime"], {i, 3}]] // Simplify;
Print["Decreasing time for deicing"]
Table[
  expectedTimeAtAirport /. { $\lambda \rightarrow 1.25$ ,  $u_{runway} \rightarrow 2$ ,  $u_{gate} \rightarrow 1$ ,  $u_{deicing} \rightarrow i$ }, {i, 1, 10}]
Print["Decreasing time for landing / takeoff"]
Table[
  expectedTimeAtAirport /. { $\lambda \rightarrow 1.25$ ,  $u_{runway} \rightarrow i$ ,  $u_{gate} \rightarrow 1$ ,  $u_{deicing} \rightarrow 1$ }, {i, 2, 10}]
Print["Decreasing time for turn around"]
Table[
  expectedTimeAtAirport /. { $\lambda \rightarrow 1.25$ ,  $u_{runway} \rightarrow 2$ ,  $u_{gate} \rightarrow i$ ,  $u_{deicing} \rightarrow 1$ }, {i, 1, 10}]

Decreasing time for deicing
Out[ ]:= {3.46154, 2.37463, 2.16897, 2.07677, 2.02369, 1.98901, 1.96452, 1.94628, 1.93216, 1.9209}

Decreasing time for landing / takeoff
Out[ ]:= {3.46154, 3.04439, 2.91808, 2.85436, 2.81525, 2.78859, 2.76915, 2.75432, 2.74261}

Decreasing time for turn around
Out[ ]:= {3.46154, 2.96154, 2.79487, 2.71154, 2.66154, 2.62821, 2.6044, 2.58654, 2.57265, 2.56154}

```

### Reducing service time:



Increasing  $\mu_{\text{deicing}}$  from 1 to 2 gives an incredible increase to performance. This is partly due to the fact that we are halving the time needed to deice, but as the performance gains flatten out when you focus on a variable, reducing time for deicing is a good way to boost performance.

Both the runways and the gate show less effect than deicing, but are still worthwhile to optimize if possible. However, both of these are hard to optimize in a real life scenario. For example, halving the time needed to turn around a plane will be close to impossible as passengers require a large enough time window to board. Therefore one should rather be trying to either optimize deicing time or increase the amount of deicing stations and / or runways.

To conclude:

The main bottleneck is the deicing stations and spending resources on either increasing capacity or reducing time needed to deice will yield good results. Do note the relationship between time needed to deice and capacity. Increasing the capacity and decreasing the time required to deice simultaneously will yield less returns than the numbers above might suggest. This is due to the fact that lowering the time per deice will also lower the probability that both of the deicing stations are in use when a new plane arrives.

As decreasing “turn around”-time, landing-time and takeoff-time are hard in a real life scenario, increasing the amount of runways to three is the best choice for increasing performance outside of cold weather.

---

## Functions and hints