Object:

We want to prove these after the experiments.

1. To let the animal grow to aimed size on aimed day, we have different feeding methods, which would cost different amount of money.
2. Different feeding methods would result different profits.
3. Unlimited feeding method would not bring the maximum profit.
4. Different r would bring different maximum profit

Parameters:

In this simulation software, there are several important parameters. They are listed below:

1. Start Day: this parameter is the date that animals are put into the feeding lot since they were born.
2. End Day: this parameter stands for the date that animal are out of the feeding lot and put onto the market.
3. Weight (Size): the weight for animals in the feeding lot.
4. Revenue: the revenue of the animal if it is put on the market
5. Feeding cost: the feeding cost of raising the animal. Using different feeding method will have different feeding cost
6. Facility cost: the facility cost, which includes any fee cost on feeding except feeding, in the feeding period
7. Opportunity cost: the value that owner of the animal will lost if they take any strategy other than unlimited feeding.
8. Discount factor: the discount factor are use in the Opportunity cost.

Math functions about animal feeding:

There are three main math functions being used to generate the basic data for feeding animals. From the start of the project, August 2013, we were using one set of these three equations, which is fitted from animal data of 20 years ago; from March 2014, we get the new set of equations from the profession, which is fitted by a new animal feeding data.

These three equations are consisted of one weight function, and two feeding rate functions. I will introduce this below.

1. Weight V.s. Date equation: this function is a function of Day to represent the maximum weight that animal could reach under unlimited feeding on that date follow nature growth curve.

First Version:

Weight = 562.959283 / (1 + EXP( (-1) \* (date - 398.095995) / 189.503378))

Second Version:

Weight = 343.822718 + 1.8290907 \* date

2. Feeding Rate V.s. Size equation: this function is a function of size to represent the maximum feeding rate for the animal with particular size could have.

First Version:

Feeding Rate = 1.004226945 + 0.0176380855 \* size

Second Version:

Feeding Rate = 4.33209573 + 0.00562646041 \* size

3. Feeding Rate V.s. Size and Date equation: this function is a function of size and date to represent the feeding rate for the animal with particular size on particular date under limited feeding condition. Since this equation is derived from a theory of animal feeding, there is no change between two version.

Equation:

Feeding Rate = a \* EXP(-0.032882286 \* time) + c

C = 0.032566835 \* S ^0.075 and a = Qmax - c

Pre-inspect

Assuming we are using same selling price for animal meat, same facility cost for one day maintaining and same r in discount function through the whole experiment.

Profit function:

Profit = Revenue - - Opportunity Cost

= Revenue – Total Food Cost – Total Facility Cost – Opportunity Cost

= Revenue - - – Opportunity Cost

Where Revenue Function is Panimal \* S \* e-rt, Panimal is the selling price of animal ($/kg), S is the size of the animal when selling it, t is the selling time.

Food Cost Function is ai \* Pfood, ai is the quantity of food fed to animal on ith day (kg), Pfood is the price of the food ($/kg).

Facility Cost Function is 1 \* Pfacility, Pfacility is the price of facility mountains for one day ($/day).

Opportunity Cost is Panimal \* S \* [], Panimal is the selling price of animal ($/kg), S is the size of the animal when selling it, t is the selling time (day), t’ is the time that reach this size on the ad lib curve (day).

By inspecting these functions, we find

1. Revenue function depends on only selling time and selling size.
2. Total facility Cost depends on only selling time.
3. Opportunity Cost depends on only selling time and selling size.
4. When we have aimed selling day and selling size, different feeding methods would not influence Revenue, Opportunity Cost and Facility Cost. In other words, On a specific

Proof of statements above:

1. It is obvious to see, since Panimal and t are constant
2. Since there is no i term in Facility Cost function, then we can simplify the summation by pulling the facility Cost function out. Then we have

Total Facility Cost = t \* Pfacility \* , t is the time animal is sold.

It is to see that Total Facility Cost is only depends on t, selling time.

1. Originally, the opportunity cost function depends on S, selling size, t, selling time, t’, the time that reach this size on the ad lib curve. However, we could using S to represent t’ by using the invert of max size function given day. After the transform, the opportunity cost function becomes only depend on size and selling time.
2. By having the statement 1,2,3, this statement is proved.

By having the above statements, we could conclude that, given specific selling size and selling time, revenue – total facility cost – opportunity would be stable. Given selling size and selling time, only variable in the profit function would be total food cost. We could list all the possible selling size and selling time pairs under constraints of max size on each day, size and feeding time boundary. Then we find the minimum food cost for the specific day and size. Using revenue – total facility cost – opportunity minus the minimum total food cost to reach this day and size would giving all the possible profit for each day size pair. Choosing the maximum in it would give the best profit through all possible result. Now we have shrunk this problem into finding the minimum total food cost to reach specific size on specific day.

By reducing this problem to find the minimum total cost problem, we could use Dijkstra’s algorithm. Dijkstra’s algorithm is a algorithm that helps to find the minimum path from start point to finish point by traversing all the node in a network. It is be sure to give the right result.

Then we want to build a graph based on the 3D plot under following rules:

1. Every possible (day, size) pair as a node in the network.
2. For (t,s) pair, it could only be connected to (t+1,s’) where s’ is [s,s’], s’ is the maximum size on the next day.
3. S’ does not necessary mean the maximum size on day t+1. If s is the interval value in the on day t, we have map it back to the adlib curve. S’ would be the max size on next day after mapping back.
4. Set the weight on the edge with value Pfood \* ai \* . Ai is bounded by maintain food amount, and reaching max size food amount (the two feeding equation).

After we have the network, we could run Dijkstra’s Algorithm to have the minimum total food cost. Using the Revenue minus these costs would give us the maximum profit.

This brings us new statements:

1. if the Pfood is constant, Pfood would not influence the choice in feeding method.
2. If the Pfood is not constant for every day, Pfood would influence the choice in feeding method
3. The discount factor would influence the choice in feeding method

Proof for the statement:

1. If Pfood is constant, we could pull Pfood out from the total food cost. Then it would not influence the choice in Dijkstra’s Algorithm

Background about Dijkastra’s Algorithm:

**Dijkstra's algorithm** is a [graph search algorithm](http://en.wikipedia.org/wiki/Graph_search_algorithm) that solves the single-source [shortest path problem](http://en.wikipedia.org/wiki/Shortest_path_problem) for a [graph](http://en.wikipedia.org/wiki/Graph_(mathematics)) with non-negative [edge](http://en.wikipedia.org/wiki/Edge_(graph_theory)) path costs, producing a [shortest path tree](http://en.wikipedia.org/wiki/Shortest_path_tree). For a given source [vertex](http://en.wikipedia.org/wiki/Vertex_(graph_theory)) (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined.

More on Dijkstra’s Algorithm:

Let the node at which we are starting be called the **initial node**. Let the **distance of node *Y*** be the distance from the**initial node** to *Y*. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the *unvisited set*consisting of all the nodes.
3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one. For example, if the current node *A*is marked with a distance of 6, and the edge connecting it with a neighbor *B* has length 2, then the distance to *B* (through*A*) will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Select the unvisited node that is marked with the smallest tentative distance, and set it as the new "current node" then go back to step 3.

## Description[[edit](http://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&action=edit&section=2" \o "Edit section: Description)]

***Note:****For ease of understanding, this discussion uses the terms****intersection****,****road****and****map****— however, formally these terms are****vertex****,****edge****and****graph****, respectively.*

Suppose you would like to find the shortest path between two [intersections](http://en.wikipedia.org/wiki/Intersection_(road)) on a city map, a starting point and a destination. The order is conceptually simple: to start, mark the distance to every intersection on the map with infinity. This is done not to imply there is an infinite distance, but to note that intersection has not yet been *visited*; some variants of this method simply leave the intersection unlabeled. Now, at each iteration, select a *current* intersection. For the first iteration, the current intersection will be the starting point and the distance to it (the intersection's label) will be zero. For subsequent iterations (after the first), the current intersection will be the closest unvisited intersection to the starting point—this will be easy to find.

From the current intersection, update the distance to every unvisited intersection that is directly connected to it. This is done by determining the sum of the distance between an unvisited intersection and the value of the current intersection, and [relabeling](http://en.wikipedia.org/wiki/Graph_labeling) the unvisited intersection with this value if it is less than its current value. In effect, the intersection is relabeled if the path to it through the current intersection is shorter than the previously known paths. To facilitate shortest path identification, in pencil, mark the road with an arrow pointing to the relabeled intersection if you label/relabel it, and erase all others pointing to it. After you have updated the distances to each [neighboring intersection](http://en.wikipedia.org/wiki/Neighbourhood_(graph_theory)), mark the current intersection as *visited* and select the unvisited intersection with lowest distance (from the starting point) – or the lowest label—as the current intersection. Nodes marked as visited are labeled with the shortest path from the starting point to it and will not be revisited or returned to.

Continue this process of updating the neighboring intersections with the shortest distances, then marking the current intersection as visited and moving onto the closest unvisited intersection until you have marked the destination as visited. Once you have marked the destination as visited (as is the case with any visited intersection) you have determined the shortest path to it, from the starting point, and can trace your way back, following the arrows in reverse.

Of note is the fact that this algorithm makes no attempt to direct "exploration" towards the destination as one might expect. Rather, the sole consideration in determining the next "current" intersection is its distance from the starting point. This algorithm, therefore "expands outward" from the starting point, interactively considering every node that is closer in terms of shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path, however, it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies.

What I impletemented:

JavaScipt:

PHP:

Python:

Refactor:

User Interface:

**Lesson learned As Team:**

**Communicate is the key to success:**

From this project, I learned that Communication is the key to success. In a group, bad communication would lead to misunderstanding, not only on technical ways, but also on the relationship between teammate.

The most time waste in a project is about the misunderstanding on some technical problems and implementing it based on the misunderstanding. If we are lucky, we will have to wait until the result comes out and see the result is different with our prediction. Then we have to take more time to tract down from our programs. After that, we have to discuss about why there is a mistake hidden in the program. The whole process will take about two times more than the usually development time. We actually run into this problem once. It was at the start of the second semester. We began to implement our algorithm in Python. Boyi were not quiet familiar with python, so the implementation job fell onto Yifang. With a small amount of time, Yifang finished his programming in python, which was based on a finished version in PHP. However, after running the program, Boyi found that the data generated from Python did not match with the prediction result at all. In the following week, the whole group was trying to find where was bug. At the end, we found that the implementation of the algorithm were totally wrong. Yifang did not understanding the algorithm right at all. And based on his misunderstanding, the software did not run as it supposed to run. As a result, another week were spent on rewriting this part.

In this case, the misunderstanding between Boyi and Yifang leads to a waste of two-week time. Both Boyi and Yifang thought they understanding each other right. However, the result turned out it is not. All of this could be improved under better communication. The way to solve this is that both sides of people could describe what they will do in detail and let the other side to agree on that, instead of just tell the other they will do this in general. Also when implementing, if confuse comes out, contact the one who could solve you question immediately, instead of telling self the understanding of his own is correct. In conclusion, good communication could save more time on software development.

Moreover, good communication could improve the relationship between teammates. Good communication could avoid some small conflicts between teammates. As a group, sometimes under pressure, the relationship between team members is fragile. However, communication in time will save it from the cliff and move the whole group quickly. Once, at the end of second semester, we had 2 weeks to finish our project. We were still working on programming the User Interface. The whole team were under huge pressure. Everyone were worried about the schedule. However, there is a job switch between Boyi and Chris. Boyi did some job of Chris without telling Chris and after that decided to continue working on this. Chris were kind of upset about what Boyi did that takes Chris’s Job without notifying Chris. After that, Boyi and Chris had a small conversation about this and agreed on Boyi was not actually mean to take Chris’ job. He just worried about the switching of job would waste more time and kept working on that part. Both of two people did not want to upset others. After that, the whole group moved on very fast and fin ish the program on time.

Communication would avoid any huge fight in time, which saves a lot of time on fighting and not satisfied with each other. Instead, team could spend these amount of time on developing software.

In a word, great communication in the group will make everything easier and more efficient.

**Do code review often:**

Code review could help the team improve their software performance as well as debug. During the code review, the whole team could find plenty of questions. Firstly, code review could help the team improve the structure of whole software. At first, the whole structure of software is always designed by one member. In our case, the structure is designed by Boyi. However, the idea of one people is limited. However, during the code review, everyone could reflects their idea and improve the whole software based on everyone’s idea. Secondly, code review could help group to debug. As we talked above, on one stage of development, we have misunderstanding on implementing the algorithm. At this time, code review saves us from hell. During the process of code review, Yifang explained every function he wrote to everyone of our group. Then during this process, we could actually read though all the code and after that, everyone could do the debug job, instead of Yifang his self. Also during the code review, we had improve some part of his code to make everything work more efficient. There are more examples like this happened during our project.   
In conclusion, code review actually makes teamwork more eaiser and more efficient. It assign one’s job to three peoples, which makes a mission more easier. Also, during the code review, we could actually see the mistakes and improve the performance, which is a win-win solution.

**Lesson learned As Individual:**

**Do not rely on your teammate:**

During the project, I learned that do not rely on your teammate, no matter how powerful your teammate is. In a project, every member should contribute to the project. Do not be lazy because your teammate is powerful and efficient, and you hope he could do all the work for the team.

**Be brave to give suggestion to others:**

Suggestion is actually a boost of the whole team. People are good at pointing out the mistakes other makes, while it is hard for themselves to see the mistakes thay are making. So it is helpful to provide suggestion to other and always be open to the suggestion that other give to you. Although sometimes it hurts, it will really help us to improve. Besides, when giving suggestion to others, please be in a nice way. It is a natural reaction that people will resist something seems hurt to them. So nice words will makes the suggestion works more efficient.

**Be motivate for your teammate:**

Everyone want to be lazy in a group and hopes other people will carry them to the end. So it is important to have someone being motivate in your group. He will push you always and get to the checkpoint in time instead of putting all missions at the end. It is really important to have this guy. Or all of you guys will be free through the semester. And then at the end semester, everyone work at late nights. Also under the pressure, most of people are not that productive and somehow, you will not reach the expectation you made at the start of the project.

**Experience will save you more time and energy:**

Experience actually is a huge factor to achieve the success. In our project, we do not have a technical person as supervisor. Most of the structure decision are made by us. And most of them lead us to failure. However, if we have the experience on this, I think we will make better decision, which will accelerate the process of developing. We actually walk in a huge circle. We start with javascript which is more toward the front end. However, the calculation power of browser are not as powerful as we think. Then we think the back end will have better calculation power than front end. We implement a version in PHP. However, PHP does not help either. After this, we start to programming for the stations, which definitely have the power to do the simulation. So let us imagine, what if we choose programming in python in the first place. We will actually save half of the time and could use it to develop more functions to make the software more friendly and more intelligent.

After the project, I know that experience is actually a treasure for the programmers. It will have to build a more stable structure for your product and saves more times in development.

**Things need to be done after:**

**Add more explanatory elements in the UI:** Based on the feedback of professors, we should add the unit for each parameter. Moreover, add help buttons for each parameters will help the users be more clear about what should be the input.

**Integrate Opportunity Cost function into the software:** For right now, we have written the opportunity function in the source code. However, we do not integrate Opportunity cost function into the software yet, because the professors have not finalize the equation for the opportunity cost.

**Integrate the outside library with the software:** There are several libraries we are currently using have dependence on other library. The only way we figure out is install one python editor that contains all these libraries. However, that should not be the solution for our software. We would spend more time on downloading the libraries and link these libraries with our program.

**Self configure product after installing:** This is the step after linking the libraries. Also, we do not which user environment this software will be under, so we should write something to do the auto configuration after installing.