For office use only
T1__________________
T2__________________
T3__________________
T4 __________________

For office use only
F1__________________
F2__________________
F3__________________
F4 __________________

2017

20th Annual High School Mathematical Contest in Modeling (HiMCM) Summary Sheet
(Please make this the first page of your electronic Solution Paper.)

Team Control Number: 8479
Problem Chosen: B

Please paste or type a summary of your results on this page. Please remember not to include the name of your school, advisor, or team members on this page.

A group of wealthy winter sport enthusiasts have asked us to develop a schematic to turn the Wasatch Peaks Ranch in Utah into a top-notch winter resort which could hold the Winter Olympics in the near future. Our task was to find optimal layouts with ski trails for all different skill levels. We were also tasked with ensuring a well-balanced set of trails, close to the 20-40-40 percent distribution for beginner, intermediate, and expert trails.

Our team mapped elevation data from Google Maps onto the range, creating a topographical graphic that demonstrated the various contour and elevation trends in the Wasatch Peaks Range. In addition, we used aspects of multivariable calculus to find gradients of the surface and use local maximums as starting points for ski trails. From there, we developed a genetic algorithm to evaluate different starting points, optimizing the efficiency of the layout. This simulation utilized natural selection-like and Monte Carlo methods, allowing random mutations to occur and increasing the replication of efficient designs. After many generations, the genetic algorithm produced a relatively optimal layout, taking into account length of trails, natural contouring of the mountain, straightness of paths, and exposure to sunlight. We then compared this proposed schematic to other North American and Olympic ski resorts to determine a ranking.

Based on our results, we have determined an extremely close to optimal resort design, taking into account all of the aspects that we were told to consider, and more. In addition, we have designated a portion of the ski slopes for possible Olympic use; that is, the collection of slopes we have designated has the full capacity to host the Winter Olympics, one of our major goals. Furthermore, the process we have developed will prove useful in the event that our clients choose to consider developing a different mountain range instead of Wasatch peaks, considering the customizability of our model.

We strongly suggest that our clients use a design that is at least similar to ours, as we have taken care to optimize it in many respects and believe it is fully capable of meeting and exceeding all expectations.

Sincerely,

Team 8479

Dear Ms. Mogul,

Thank you for choosing us to help design your clients' ski resort. Given your specifications and desires for the resort, we created a model to iterate through all the possible ski slope designs, and in the end, we came up with the design that will be the most efficient, provide the most fun, and feature a strong balance of beginner, intermediate, and expert level courses.

In this model, we created various topological color maps and models to better visualize the property's changes in elevation. Particular aspects we prioritized were the flat spaces for beginning and ending ski trails. As skiers need to be able to prepare themselves at the beginning, and slow down at the end, using flat spots for these components gives skiers more flexibility and helps to reduce injuries, creating a safer and more friendly skiing experience for all.

Our final design is as follows (the difficulty distribution is: Beginner - 18%, Intermediate - 36%, and Expert - 46%).

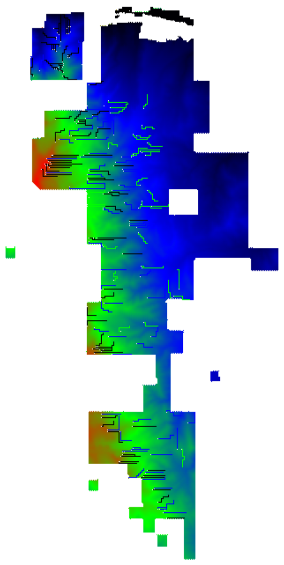


Figure 1: Our optimal design. The white points represent the tops of the slopes, while the black/blue/green layout represents difficulty (advanced/intermediate/beginner). Between the bottoms and tops of slopes of comparable elevation, small trails and runs will be constructed. The background color gradient is a topological graph running from the highest elevations (red) to the lowest elevations (blue). In addition, 25 km of cross country trails will be constructed in the blue and black areas on the top-right. For a detailed list of exactly where all of our slopes need to be built, please reference Appendix B.

We determined this to be the optimal layout by using a rating system that emphasized the length of the slopes, their relative straightness (allowing a few turns), and the difficulty ratios of the slopes. All of these combined to give us a single number, resulting in our best final design. However, there are some other possible designs which we have generated, which we have demonstrated in the the Results section.

We then designed and implemented methods to compare ski resorts, with the goal of eventually comparing our design to both North American ski resorts and Winter Olympics hosts. By analyzing correlations between `skidesign.info` ratings and various factors taken from data that we found, we were able to select four key factors - peak height, vertical drop, skiable acreage, and total trail length - that we could use to compare our proposed resort to other North American resorts. To design our method, we used our data to create Z-scores for each of these four categories for all of the resorts in our data set, thus allowing us to create a single score (the sum of Z-scores) for each ski resort.

While using this method to compare our proposed design to various other North American resorts as well as previous Winter Olympics hosts, we found that our resort would without a doubt rank in the top 5 of North American resorts and compare very favorably to those of other Olympic hosts. With an impressive 5500 acres of ski-able land and over 160 km of trails and slopes of all difficulties, it became clear in our analysis that the prospective Wasatch Range Resort - with our design and your dedication - will be one of the best in the world.

Sincerely,
Team 8479

The Wasatch Range Ski-Matic

Team 8479

November 2017

Contents

1 Introduction

Ski resorts are very popular among both avid skiers and families who are looking to have fun. In fact, winter resorts account for over 3 billion dollars in revenue, with an annual growth of 2.6% [IBI]. In addition to their use as recreational areas, ski resorts are also used for various winter competitions, most famously the Olympics. Our job was to design and create a suitable ski resort for both the Olympics and for recreational use in the Wasatch Peaks Range property. In addition, we were advised to create over 160km of trails and maintain a 20%-40%-40% balance between beginner, intermediate, and expert trails. With these conditions and the information on the Wasatch Peaks Range, we created our model.

Ski resorts, like all large-scale building projects, are subject to a variety of factors that have to be considered. Not only do top-quality ski resorts have to cater to casual families and skiers, they must also be able to satiate the needs of the more advanced and skilled who would quickly grow bored of beginner slopes. In addition, this latter consideration correlates greatly with the ability for a resort to host high-level competitions, including national and international competitions, especially the Olympics.

Our model attempted to encompass all of these factors, in order to create the best possible ski resort. Firstly, we extrapolated the outline of the property from the map found online. Combining this with a few landmark coordinates, we were able to determine the coordinates that each pixel in the outline picture corresponded to. Then, using the Google Maps API, we requested the various elevations of these pixels and stored them. From this data, our model determined the possible paths of each difficulty and using a ranking algorithm, we were able to determine high-quality ski trails to use. We then used sophisticated statistical analysis to compare our proposed design to other ski resorts over a variety of factors.

2 Assumptions

We began with a series of assumptions to facilitate our model and allow ourselves the best possible conditions for creating the optimal ski resort.

Assumption 1: The cost of building the ski resort is not a concern.

Justification: Ski resorts are known to be very expensive; several resorts have sold for tens of millions of dollars.[Spi] As a result, the group of investors and Ms. Mogul should be well aware of the high costs for building an Olympic-level ski resort. In addition, they have laid out the aims of the project (over 160 km of slopes) and given the area for usage, so they should also be aware of the scope of the resort. Since this project should not run into any extra problems, its price will match other resorts of the same size.

Assumption 2: Longitude and Latitude can be assumed to act linearly (essentially, the earth can be considered to be flat with respect to this area).

Justification: Longitudinal and latitudinal deviations from flatness are very small, especially when focusing on specific areas, as this model entails.

Assumption 3: Snow cover should be adequate for skiing and have equal depth across all regions of the mountains.

Justification: If conditions for snow were not met in the winter (the time of peak interest that our customers want), there would be no desire for a ski slope as all the snow would melt. So, there must be snow-favorable conditions, at least in winter. If needed, snow-making machines can be used to add to the snow cover, allowing freedom in designing trails. Additionally, although there would be variations in the depth of the snow between different regions, in practice these will not be large or common enough to warrant treating them differently, and as such we can assume that snow cover will add an equal amount of elevation to all parts of our ski course.

Assumption 4: There are no restricted areas on which trails can be built on the Wasatch Peaks Range property.

Justification: If this were the case, then the developers and customers would have let us know. In addition, part of the costs of developing ski trails is clearing the needed land, so the presence of trees and other obstacles should not impede development by significant measures.

Assumption 5: Relatively small stretches of slope (26) are linear in nature. That is, there are no extreme jumps or changes in elevation between two points separated by 26 meters.

Justification: The distances between our data points are fairly small, and as such would not allow room for non-negligible variances in elevation beyond a smooth linear elevation change from one data point to the next. Therefore, we can assume that such variance would not be significant enough to factor into our model, and as such we can assume our slope will be smooth and linearly decreasing in elevation between data points. In addition, any severe discrepancies can be resolved by using snow to even out the differences.

Assumption 6: The distance covered by 1 degree of latitude is 69 miles. The distance covered by 1 degree of longitude is 53 miles.

Justification: The distances between degrees of latitude ranges from 68.7 miles to 69.4 miles (due to the earth's ellipsoid nature). However, these numbers are close enough and our area of interest is sufficiently negligible that we can assume it is 69. Meanwhile, longitude changes much more, with its maximum at the equator and minimum at the poles. However, at around 40 degrees North (which is roughly the longitude of the region the ski resort will be built in) the distance is approximately 53 miles. [Qay]

Assumption 7: The elevations of specific points provided by our data will not change throughout the year.

Justification: Although weather, natural processes, and human activity can all slightly change the elevation of points throughout the year, many of these changes will not cause major changes to the elevations of specific points. Just as importantly, it will not change the elevation of all points in our region fairly evenly, thus preserving relative elevations between points.

Assumption 8: One pixel corresponds to a square area of 26 meters by 26 meters.

Justification: When calculating the pixel length and width, using the values we obtained in 3.2, longitude and latitude gave pixel sizes of 25 and 27 meters, respectively. Since these are pretty close, and single meters are relatively negligible on such a grand scope, we can safely assume the 26 by 26 shape.

Assumption 9: Ski slopes should start from peaks (high points) and end at valleys (low points).

Justification: This would maximize the area for skiing, leading to more and longer ski trails. In addition, skiing is obviously downhill, so starting from peaks and ending at valleys allows us to simplify the problem slightly.

3 Model

Note: See Appendix A for all referenced code.

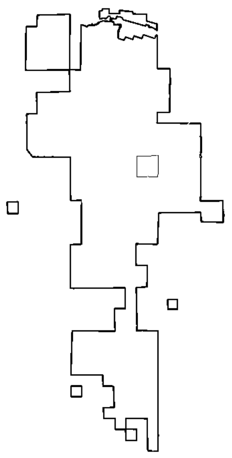
3.1 Goals of Model

Using our mathematical model, we seek to:

1. Model the terrain of the Wasatch Peaks Range and create a topological gradient to accurately show this. Also, determine the elevations across the terrain.
2. Analyze possible designs for over 160 km of ski slopes and trails and determine the most optimal and attractive design, making sure to keep in mind the demands of an Olympic-ready course and the 20-40-40 split of beginner, intermediate, and advanced slopes.
3. Design metrics to evaluate the effectiveness of ski various resort designs and using them both to compare candidate designs for our ski resort and to compare our proposed resort with other ski resorts over various categories.

3.2 Creating the Outline of the Property

We used a Python script to trace the topographical map provided to us by Ms. Mogul and find the geographical boundaries of land provided to build the ski resort. With this script, we were then able to create an "outline" of the Wasatch Range territory land that we would be filling in with data, as shown below:



3.3 Finding Elevation and Creating a Topological Color Map

To determine the elevation of distinct points, we used the Google Maps API (`https://developers.google.com/maps/`). We stored the elevations of 18,600 points, evenly spread throughout the property. This corresponds to one pixel for every four. In order to determine the latitude and longitude of each pixel (which was required for the Google Maps API), we used points given on the map found at the Wasatch Peak Range website, `www.mirrranchgroup.com/ranches/wasatch-peaks-ranch/#prop-maps`:

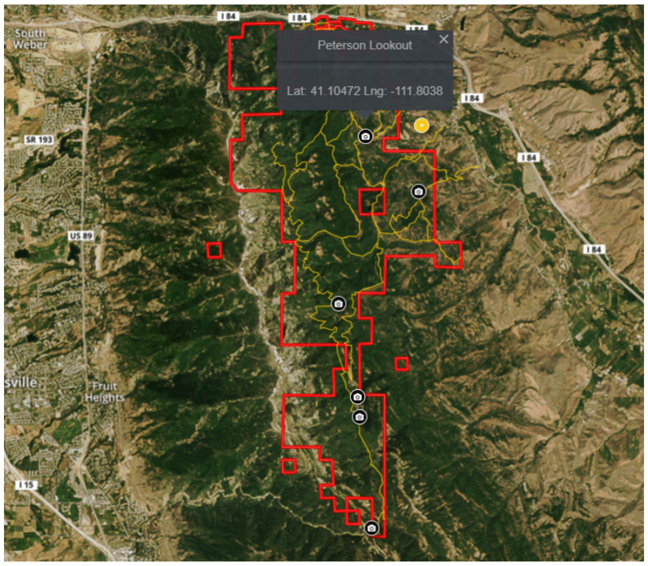


Figure 2: Online map with landmarks and their coordinates.

As the maps gave us the longitude and latitude, and we had assumed linearity, we could determine the pixel scale to longitude and latitude. Graphing and determining the line of best fit, we determined that each pixel corresponded to 0.0003241 degrees of longitude and 0.0002346 degrees of latitude. Then, each pixel (x, y) had a latitude and longitude of

$$(-0.0002346y + 41.165°, 0.0003241x - 111.915°) .$$

We were then able to store these elevations in array form. Gaps in the data (from only taking the altitude of one out of every four pixels) were filled in by assigning to each ”missing” pixel an elevation that was the average of the elevation of the four pixels adjacent to the missing one. We determined that taking the average of the elevation was the most accurate way, as this gave better results than any other regression, including a polynomial regression:

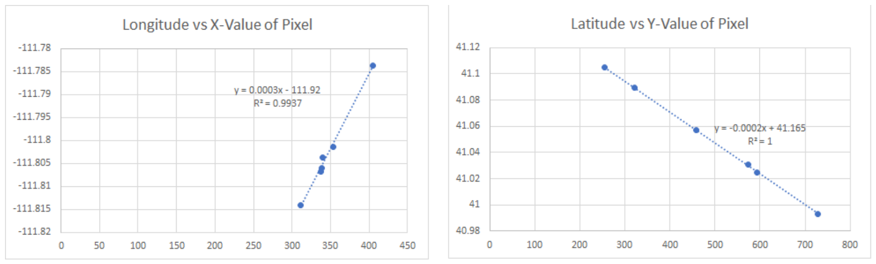


Figure 3: The charts used to determine the linear regression. More significant figures were obtained by multiplying the y-values by 1000, then normalizing back afterwards.

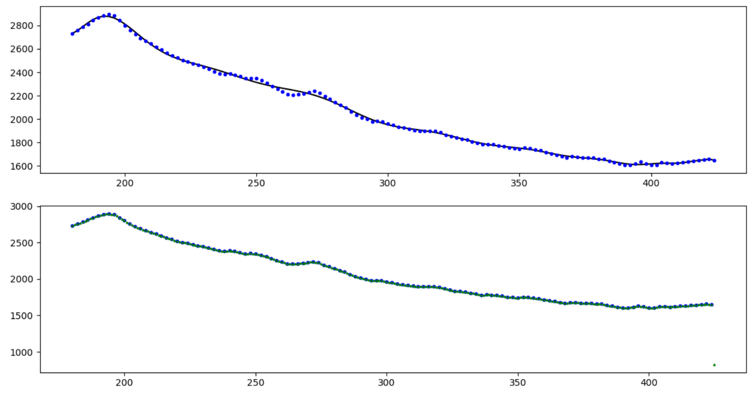


Figure 4: Slice of the elevation graph at slice $y = 290$. The line in the top graph is polynomial regression, while the bottom graph uses the average of the neighboring points. It is easy to see the latter is more accurate.

Additionally, by writing a Python script to assign a color to each elevation in the possible range of elevations for Wasatch Peaks, we were able to create a topological map to visualize the data collected.

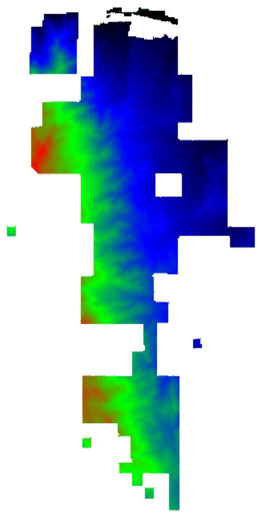


Figure 5: Gradient map generated using Google Maps API elevation data. Red points are highest altitude, while dark blue points are lowest altitude.

3.4 Using the Gradient to find Peaks and Valleys

In order to calculate the rate of change of the elevation of the surface, we calculated the gradient at each point. The gradient at a point in the xyz plane is defined as:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$\frac{\partial f}{\partial x}$, or the partial derivative of f with respect to x, represents the change in altitude as x changes while holding y constant. $\frac{\partial f}{\partial y}$ represents the change in altitude as y changes while holding x constant. If both $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y} = 0$, the altitude levels out at that point. At points on the surface where the surface is flat, $\nabla f = 0$, so we knew that these were either peaks, valleys, or flat points. We used the python function *numpy.gradient* to estimate the gradient at each pixel on our outline. After testing neighboring points to split them up into maxes and mins (disregarding the flat points in the middle), we used these for our start and end points, respectively, and as a result, we were able to create paths between these two sets.

3.5 Generating Potential Slopes

Using the peaks determined with the gradient, we found the possible slopes with these as the sources by comparing the elevations of the surrounding areas and comparing them to our peaks. We continued these trails down until they could not be continued. In addition, we classified these trails as beginner, intermediate, and expert based on their percent slope (which we also determined from the gradient). This allowed us to determine all the potential paths that went down from specific peaks in the area.

One example of the graph of beginner paths is shown below, in which the white squares show the peaks and the black squares stem off of the peaks.

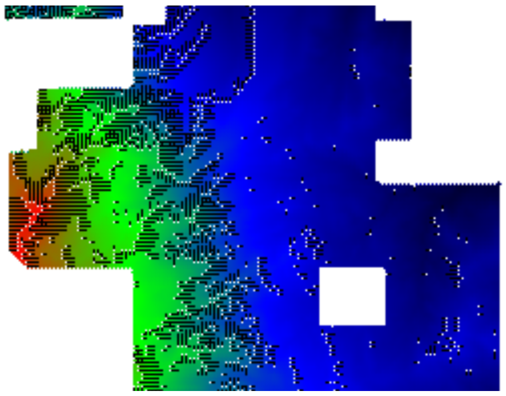


Figure 6: A section of the property showing the possible expert trails from our 'peaks.' These trails range in size, but our next step dealt with limiting the trail lengths

Afterwards, we determined the paths that were sufficiently long to allow for ski slopes to be formed. Normal ski paths are usually supposed to be around 650 to 1200 meters long (per the International Federation of Skiing Rules) in order for races to occur. As we intend this resort to be Olympics-ready, we would like to make all of the slopes Olympics level. So, we isolated the paths that were 20 to 60 pixels long, which gives numbers close to the desired range.

It is also important to note that most ski trails are 45-50 meters, or around 2 pixels, long. Our paths, although only 1 pixel wide, can be easily expanded $\frac{1}{2}$ a pixel on either side to meet the width requirements, as desired.

3.6 Designing Metrics to Compare Slopes

To compare slopes, we identified and considered four main factors: **length** (the total length of each ski slope), **fall-line analysis** (how well ski slopes follow the natural contouring of the mountain), **curviness** (how much the slope curves as it moves down the mountain), and **north-easternness** (as the optimal location of ski slopes is on the northern or eastern side of a mountain, the ideal ski slope would face north or east).

We then designed metrics for each of the four slope comparison factors. For **length**, we simply computed the distance (in pixels, while taking elevation converted from meters into pixels (26-meter units) into account) between the start and end points. For **fall-line**, we considered each downward "step" that the path took from a pixel to a lower pixel. If, at any of these steps, the ski slope could have gone to an even lower pixel in elevation, indicating that the slope would be following the natural contouring of the landscape as well, we considered the step "suboptimal". However, if the slope did follow the natural contouring of the landscape by going to the lowest possible point, we considered the step "optimal". Our final contour metric used was the fraction of all steps in the path that were optimal. For **curviness**, we took the slope's endpoints, found the Pythagorean distance between them (in pixels), and then found the total number of steps the path took. We then divided this total number by the first distance to create a metric that would assign lower values to straighter paths (with the lowest value of 1 indicating that the number of one-pixel steps taken equalled the distance in pixels, or a straight line) and higher values to curvier ones. Finally, for **north-easternness**, we simply added the total change in the x and y coordinates of the path towards the east and north, and divided by the path length.

For contouring and north-easternness, a bigger number would be better, as this would represent paths that faced closer to the north-east and paths that followed the natural landscape better. For path length, numbers closer to 1000 meters (or roughly 38.46 pixels), the optimal slope length as noted by the Federation of International Skiing would be considered better would be considered better, while for curviness, numbers closer to the mean of that type of level would be better. It then remained to find an equal weighting of our four factors (contouring, north-easterness, (length's distance from 38.46 pixels), (curviness's distance from the mean of that level). To achieve this, we computed the standard deviation of these four factors. For each slope, we then computed the z-score of each of the four factors in the slope (the z-score being the total distance from a mean divided by the standard deviation of that factor) and created a weighted-average of the z-scores to achieve a total score.

Using weightings of 40% for length, 30% for contouring, and 15% for curviness and north-easterness, we were then able to create a single formula to compute a Slope Index formula for each type of slope (beginner, intermediate, or expert). If we wanted to compare two slopes, we could then simply use the formula and take the one with the higher Slope Index (comparisons across types being valid due to the z-scoring present in the formulas).

3.7 Generating a Graph Describing Where it is Possible to Travel to From Each Pixel

In order to properly generate the possible paths, we first used a program called DescentGraph that stored four values for each coordinate describing the directions that each pixel could descend into. Whether or not a point could descend into another point was determined by the slope, given by the formula $\frac{\delta Elevation}{26}$ (where division by 26 was necessary to convert from meters to pixels), from the original point to the possible point to descend into. This program also generated the starting points of the paths to more efficiently calculate the longest paths. By compiling this list, we were able to quickly generate the possible paths for different difficulty levels.

3.8 Generating possible paths for each difficulty

Using the compiled graph from the DescentGraph program, we calculated every possible path from each starting point for specific difficulty levels with a program called LongestPaths. This used a Depth First Search algorithm, a graph search algorithm that allowed us to create a path from each starting point using recursion to continue along a path until it couldn't continue. We programmed this algorithm to follow paths along the way, allowing us to store the paths as a list of the pixels it went through. Since we wanted the slopes to be at least a certain length, this algorithm then added only the paths that met a length requirement to a list of 'long' paths that could be used in the Genetic Algorithm.

3.9 Using a Genetic Algorithm to Create Potential Resorts

A genetic algorithm seeks to model the process of natural selection by weighing the results of several generations of modifications and producing the best, or most fit, design. We started with four layouts with randomly generated starting points for paths. Each layout was assigned a fitness value based on length of trails, fall-line analysis, curviness of of trails, and direction. After each generation, the layouts with the two highest fitness values "survived," while the other 13 layouts were replaced with mutations of the best two. Although each layout in the first generation used 4 starting points, mutations resulted in variations in both the number and location of starting points. This process continued for 200 generations.

Every mutation randomly generated 20 more slopes of the same difficulty to consider. For each slope, if it did not intersect with any of the existing slopes and did not start too close to another slope, then it was simply added. If it did intersect or start to close to one of the other slopes, then the new slope was added and the old slope was removed. In this way, we were able to continuously consider many different paths in each generation while efficiently improving with each mutation and each generation.

Our genetic algorithm first added expert slopes, then intermediate slopes, then beginner slopes. In this way, we prioritized good expert slopes and then good intermediate slopes, which would be more important for an Olympic event. Additionally, because there were many more possible intermediate slopes than expert slopes and beginner slopes than intermediate slopes, running the genetic algorithm in this order allowed us to have more options for the different slope difficulty levels. Running all difficulty levels of the slopes into the genetic algorithm simultaneously would not only take much more time, but would also be more inconsistent and could compromise some good expert, which are more important. Additionally, since we had few possible expert slopes, this was much better and more consistent in meeting the 40% expert slope requirement. The genetic algorithm returns a list of all of the paths that would be included in a ski scheme, with each path being represented by a list of pixels that the continuous path was comprised of.

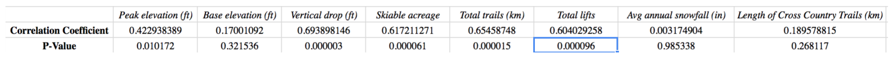
4 Comparing Designs and Existing Ski Resorts

4.1 Data Collection

To collect data, we began with the SkiSlopeComparison.xls data provided. We then scanned a random selection of thirty-six of the North American ski resorts and used the pre-provided data of **peak elevation, base elevation, vertical drop, skiable acreage, total length of slopes, total lifts**, and **average annual snowfall**. We then pulled from the website skiresorts.info a list of star ratings for these thirty-six North American ski resorts, as well as the total length (in kilometers) of cross-country/Nordic ski trails. The data was collected in a Google Sheets spreadsheet.

4.2 Parameter Consideration

After the data was collected, we were then able to calculate correlation coefficients between average rating (the dependent variable) and all of our other data (independent variables). Once the correlation coefficients were calculated, we fed the correlation coefficients into VassarStats' online tool to convert our coefficients and sample size ($N = 36$) into a P-value in order to determine which of our potential factors actually influenced the `www.skiresort.info` star ratings of the resorts. A table of our resulting correlation coefficients and P-values is shown below.



	Peak elevation (ft)	*Base elevation (ft)*	*Vertical drop (ft)*	*Skiable acreage*	*Total trails (km)*	*Total lifts*	*Avg annual snowfall (in)*	*Length of Cross Country Trails (km)*
Correlation Coefficient	0.422938389	0.17001092	0.693898146	0.617211271	0.65458748	0.604029258	0.003174904	0.189578815
P-Value	0.010172	0.321536	0.000003	0.000061	0.000015	0.000096	0.985338	0.268117

Figure 7: Table displaying correlation coefficients and P-values for relationships between skidesign online rating and a number of other factors.

Based on this analysis, we noted that peak elevation, vertical drop, skiable acreage, total length of slopes and trails, and total lifts had a significant ($P < 0.05$) correlation with the skiresort.info star ratings. However, average annual snowfall and base elevation both did not have a statistically significant ($P > 0.05$) correlation with star ratings.

Of these five significantly correlated factors, we chose not to consider total lifts. This is because, from our assumptions, the Watsach Range developers will have significant resources to the point where money and labor will not be a concern. As such, they should be able to build any number and arrangement of lifts necessary to allow the future Wasatch Range peaks resort to succeed. Therefore, our method to compare our design to other North American ski slopes only considered the four factors **peak elevation (in feet), vertical drop (in feet), skiable acreage (in acres),** and **total length of slopes (in kilometers)**.

4.3 The Main Comparison Method Used

Now equipped with our data and our factors to consider, we were able to create a method to compare our ski resort to other ski resorts. This method was similar to the previous method used to compare individual ski slopes (although with different factors); however, we did not weight the factors. By computing z-scores for each of the four factors for each of our thirty-seven (now including the proposed Wasatch Range Resort) resorts and calculating a total score for each ski resort by summing up the z-scores of its four factors, we were able to calculate a single score for any resort that could be used to evaluate it. We could then sort our resorts in descending order by this score to create a ranking of the thirty-seven resorts and evaluate how well our proposed resort would do.

5 Model Results

5.1 Optimal Resort Design

By running our genetic algorithm many times, we developed a couple high ranking resort designs that we believe will be the most beneficial to create. These are demonstrated below, with the white dots representing tops of slopes and the black/blue/green color designation signifying the difficulty (advanced/intermediate/beginner). These were produced after 200-iteration genetic algorithm was repeated many times. These models reflect our emphasis on a delicate spread between the difficulty levels (in a 20-40-40 ratio) and our want for spread-out trails. Our actual difficulty distribution was in a 18-36-46 spread, which is fairly close to the allotted ratio.

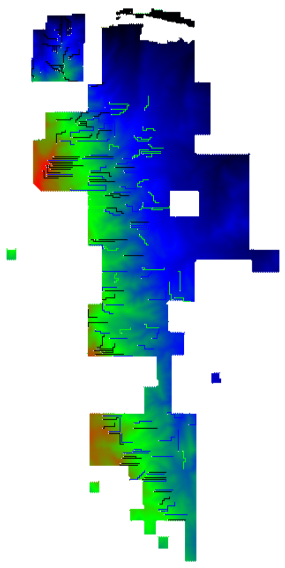
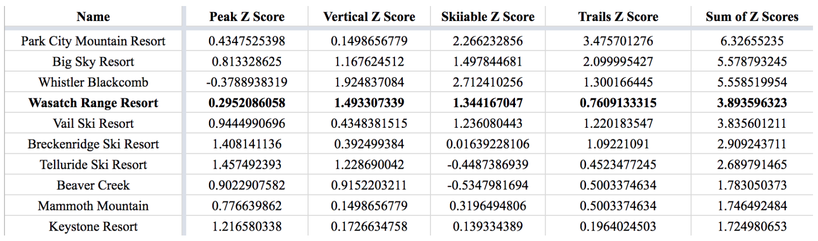


Figure 8: Our optimal design. The white points represent the tops of the slopes, while the black/blue/green layout represents difficulty (advanced/intermediate/beginner). Between the bottoms and tops of slopes of comparable elevation, small trails and runs will be constructed. The background color gradient is a topological graph running from the highest elevations (red) to the lowest elevations (blue). In addition, 25 km of cross country trails will be constructed in the blue and black areas on the top-right.

5.2 Comparison to other North American Ski Resorts

Out of the thirty-seven ski resorts provided, our comparison method ranked the proposed Wasatch Range Resort in fourth place. Although the proposed resort ranked behind Park City Mountain Resort in Utah, Big Sky Resort in Montana, and Whistler Blackcomb in British Columbia, the sum of its z-scores (approximately 3.89 - corresponding to each of its four factors being one standard deviation above the mean, on average) outperformed those of all other ski resorts in the data - including Vail and Telluride Resorts in Colorado, who were ranked second- and third-best in North America by skiresort.info. The Z-score of its peak height was approximately 0.295, and the Z score of its vertical drop was approximately 1.493, indicating that these two factors were 0.295 standard deviations and 1.493 standard deviations above average, respectively. These two factors, although strengths of ours, were essentially given to us by the geography of the region. However, the factors that we *could* significantly influence, the skiable acreage and trail length, also impressed. Our proposed resort's skiable acreage was 1.344 standard deviations above the mean, and its total trail length was 0.761 standard deviations above of the mean. Even when the first two factors were disregarded, in terms of the sum of the Z-scores for skiable acreage and trail length, our proposal still ranked fifth of thirty-seven.



Name	Peak Z Score	Vertical Z Score	Skiiable Z Score	Trails Z Score	Sum of Z Scores
Park City Mountain Resort	0.4347525398	0.1498656779	2.266232856	3.475701276	6.32655235
Big Sky Resort	0.813328625	1.167624512	1.497844681	2.099995427	5.578793245
Whistler Blackcomb	-0.3788938319	1.924837084	2.712410256	1.300166445	5.558519954
Wasatch Range Resort	**0.2952086058**	**1.493307339**	**1.344167047**	**0.7609133315**	**3.893596323**
Vail Ski Resort	0.9444990696	0.4348381515	1.236080443	1.220183547	3.835601211
Breckenridge Ski Resort	1.408141136	0.392499384	0.01639228106	1.09221091	2.909243711
Telluride Ski Resort	1.457492393	1.228690042	-0.4487386939	0.4523477245	2.689791465
Beaver Creek	0.9022907582	0.9152203211	-0.5347981694	0.5003374634	1.783050373
Mammoth Mountain	0.776639862	0.1498656779	0.3196494806	0.5003374634	1.746492484
Keystone Resort	1.216580338	0.1726634758	0.139334389	0.1964024503	1.724980653

Figure 9: The top ten ski resorts in our data set and their Z-scores in each factor, sorted by the sum of all four Z-scores.

This implications of this analysis are then clear: our proposed resort would be able to name itself among the best in North America, as it offers both the impressive natural landscape the region already possesses (the Wasatch Range's peak height and vertical drop) and a large selection of trails (created as a result of our model). If the Ski Resort is able to be built to our specifications, it can count on being one of the best in the world.

5.3 Winter Olympics Site

In our models, we noticed a constant high concentration of advanced-level ski slopes in the top-left square, disconnected from the main body. We propose this square should be the main Olympic venue to propose because of its high level courses that are all near each other.

Previous Winter Olympic sites, including the 2014 Sochi, Russia course and 2018's PyeongChang, South Korea courses are relatively small, featuring around 20 km of advanced trails in each country. This matches well with our proposed Olympic area because it is relatively small and in our case, features around 20 km of slopes, all of which are advanced. It is reasonable to assume that Olympic level skiers will not be attempting any beginner or intermediate courses, so the other level slopes in Sochi and PyeongChang don't really matter. In this way, Wasatch Range Resort will have the perfect facilities for hosting the Winter Olympics, something that may come sooner than later.

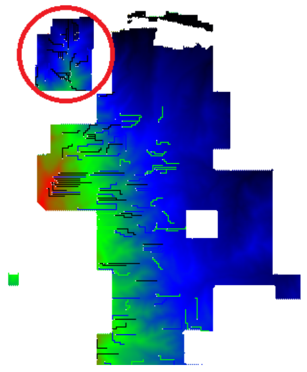


Figure 10: Our proposed location for the Winter Olympics Events.

6 Discussion

6.1 Model Implications

6.1.1 Water Flow

Through the process of generating a model of the Wasatch Peaks Resort, we demonstrated that we can use geographical elevation data to accurately calculate slopes and optimize paths across the surface. This procedure has many real-world applications. By calculating the gradient and rate of change of a surface, we can model concepts such as water flow. We can also predict the movement of glaciers over mountains.

6.1.2 Consumer Convenience and Efficiency

We also selected ski trails based on convenience for the skier. We wanted to minimize the distance skiers had to travel to move from one slope to another. We also grouped slopes of similar difficulty, considering that a beginning skier would not likely use an advanced trail. This method of testing arrangements based on convenience is also applicable in situations where the distance between destinations frequented by the same type of people need to be minimized. For example, an airport could utilize a similar method to minimize the distance between gates with connecting flights, and an amusement park could place rides of similar thrill level near each other.

6.2 Customization

Because of the way our model works, it is extremely customizable to various mountain ranges and for various purposes. The only specific information we required were the coordinates of each pixel, which can be easily computed with a few landmarks and assuming linearity. We could easily have drawn data for a different mountain range, with different amounts of beginner, intermediate, and advanced trails, and specify different criteria, and our model would still have produced the best layout of trails according to those specifications.

6.3 Strengths and Weaknesses

6.3.1 Strengths

1. Our initial data was very accurate, considering we drew almost exact elevation data from the Google Maps API. We retrieved data from over 18,600 evenly spaced points throughout the party, resulting in a relatively accurate set of elevation statistics throughout.

2. There was a lot of space at the bottom of the slope with very level elevation. We optimized this space by using it as space for cross-country skiing, since cross-country ski trails are flat, round, and don't require ski lifts.

6.3.2 Weaknesses

1. Running our algorithm for every pixel on our outline would take too long, so we averaged the elevations of every four pixels.

2. Our algorithm found paths by searching for changes in altitude between pixels. However, this meant that slight differences in altitude would be marked as very short beginner paths. Since there were so many slight changes in altitude, our algorithm found too many short beginner paths, and took too long to run. Thus, we had to restrict the algorithm to only detect beginner paths with a steep enough slope, above a minimum altitude.

7 Conclusion

We derived from our model and the genetic algorithms a ski resort design that is very close to optimal, which we have described in this report. The algorithms in the model have ensured that we have created a well-spaced array of beginner, intermediate, and advanced ski slopes, all three of which were present in the desired proportions, and the comparisons behind the algorithm have ensured that our assortment of ski slopes will be appropriately curved, sized, and located according to our specifications. Not only have we ensured that our model has created sufficient ski slopes and trails, but the design of the land has resulted in a smaller region of difficult ski slopes that would be well-suited to hosting a future Winter Olympics. By creating and refining statistical methods to analyze the effectiveness of our eventual proposed design and compare it to other North American ski resorts, we were able to demonstrate the strength of our proposed design for such a ski resort.

We therefore recommend that you begin to plan towards construction of a Wasatch Range Ski Resort according to our specifications as soon as possible. The process of construction will be difficult and expensive. However, our models have ensured that when you have finished building a ski resort with our network of trails, you will be rewarded with one of the finest ski resorts in North America and a suitable host to the skiing events in a future Winter Olympics.

Bibliography

[IBI] IBISWorld. *Ski and Snowboard Resorts in the US: Market Research Report.*

[Qay] Abdullah Qayum. *Calculating Longitude Length in Miles?*

[Spi] Benjamin Spillman. *Want your own ski resort? It'll cost you big bucks.*

8 Appendix A

Below we have attached the Python code for our model, with comments provided for understanding.

8.1 Create Outline

```
import PIL
import PIL.Image
import io
import numpy

'''
Program that retrieves the outline of the boundaries
of the ski resort from a map of the ski resort, which
will be easier to work with in other programs.
'''
img = PIL.Image.open("images2/aerialmap.png")
img = img.convert('RGB')
pix = img.load()
w, h = img.size
for x in range(w):
    for y in range(h):
        r, g, b = pix[x,y]
        #If the pixel is red, which means that it is a boundary,
        #turn it black
        if(r > 220):
            pix[x,y] = (0,0,0)
        #Else, turn the pixel white
        else:
            pix[x,y] = (255,255,255)
'''
Fill in the outparcel with black, which cannot be used
for the ski resort
'''
for x in range(335,367):
    for y in range(319,348):
        pix[x,y] = (0,0,0)

'''
Afterwards, we used Microsoft Paint and filled in
the area outside of the boundaries with black so that
white represented the ski resort area
'''

img.show()
img.save("images2/outlinemap.png")
```

8.2 Get Elevations

```
import PIL
import PIL.Image
import io
import json
import urllib.request as urllib
import math
import pickle
import numpy as np

def getelevation(lat, lng):
    apikey = "AIzaSyCmLtIDYD1nVtnQuSw8KBeilcZW8Evnuzk"
    url = "https://maps.googleapis.com/maps/api/elevation/json"
    request = urllib.urlopen(url+"?locations="+str(lat)+","+str(lng)+"&key="+api
    str_response = request.readall().decode('utf-8')
    results = json.loads(str_response)
    elevation = results['results'][0]['elevation']
    return elevation

img = PIL.Image.open("images/map2.png")
img = img.convert('RGB')
w, h = img.size
pix = img.load()
elevationlist = np.zeros((619,801))
x = 0
while (x < 618):
    y = 0
    while (y < 800):
        r, g, b = pix[x,y]
        if(r > 100):
            lat = 41.165-0.0002346*y
            lng = 0.0003241*x-111.915
            elevation = getelevation(lat,lng)
            elevationlist[x][y] = elevation
            if(elevation < 2200):
                pix[x,y] = (255,255,255)
            else:
                pix[x,y] = (int(math.floor(2800-elevation)), int(math.floor(2500-
            print (x, y, elevation)
        y = y + 2
    x = x + 2
```

```
with open("test.txt", "wb") as fp:
    pickle.dump(elevationlist, fp)

print(elevationlist)
img.show()
img.save("images/topogradient.png")
```

8.3 Create Gradient

```
import PIL
import PIL.Image
import io
import json
import urllib.request as urllib
import math
import pickle
import numpy as np

'''
Function used to generate a gradient map based on topography and
interpolates the elevations of points for which values were not
retrieved from the Google API
'''

def elevation_to_color(x):
    '''
    x: integer from 1468 to 2917 representing an elevation (in meters).
    output: an rgb that can be used to create a gradient topographical map.
    '''
    if 1468 < x <= 1951:
        return (0,0,int(round(256*(x-1468)/483)))
    elif 1951 < x <= 2434:
        return ((0, int(round(256*(x-1951)/483)), int(round(256-(256*(x-1951)/48
    elif 2434 < x < 2917:
        return ((int(round(256*(x-2434)/483)), int(round(256-(256*(x-2434)/483))

img = PIL.Image.open("images/outlinemap.png")
img = img.convert('RGB')
w, h = img.size
pix = img.load()
elevationlist = np.zeros((619,801))
tobecolored = []
elevation = 0

#Load elevation list with values for only even coordinates
```

```
with open("elevationlist.txt", "rb") as fp:
    elevationlist = pickle.load(fp)

for x in range (618):
    for y in range (800):
        r, g, b = pix[x,y]
        #Only do this for white pixels, which represent the area
        #of the ski resort
        if(r > 100):
            lat = 41.165-0.0002346*y
            lng = 0.0003241*x-111.915
            #Depeding on the coordinates, set the elevation of
            #the coordinate to an average the the elevations around it
            if(x % 2 == 0 and y % 2 == 0):
                elevation = elevationlist[x][y]
            elif(x % 2 == 0 and y % 2  == 1):
                elevation = (elevationlist[x][y+1]+elevationlist[x][y-1])/2
                elevationlist[x][y] = elevation
            elif(x % 2 == 1 and y % 2 == 0):
                elevation = (elevationlist[x+1][y]+elevationlist[x-1][y])/2
                elevationlist[x][y] = elevation
            else:
                #Find elevations of these later because points around
                #it may not have been evaluated yet
                tobecolored.append([x,y])

            print(elevation_to_color(elevation))
            color = elevation_to_color(elevation)

            if(color != None):
                pix[x,y] = color
            else:
                pix[x,y] = (255,255,255)
        #Turn the area not included in the ski resort white, as our gradient
        #used values that approached black
        else:
            pix[x,y] = (255,255,255)

#Color and find elevation of the rest of the pixels
for pixel in tobecolored:
    elevation = (elevationlist[x+1][y]+elevationlist[x-1][y])/2
    elevationlist[x][y] = elevation
    color = elevation_to_color(elevation)

    if(color != None):
        pix[x,y] = color
    else:
        pix[x,y] = (255,255,255)

#Save the elevation and topography color gradient
with open("elevationlist2.txt","wb") as f:
    pickle.dump(elevationlist, f)


img.show()
img.save("images/topogradient2.png")
```

8.4 Descent Graph

```
import pickle
import math
import numpy as np
import PIL
import PIL.Image

#Load elevations
with open("elevationlist2.txt", "rb") as fp:
    elevation = pickle.load(fp)

img = PIL.Image.open("images/topogradient2.png")
img = img.convert('RGB')
w, h = img.size
pix = img.load()

#Initiate the graph
graph = np.zeros((618,800,4))
#Slope parameters, which correspond to difficulty
minslope = 0.09
maxslope = 0.25
#b is beginner, i is intermediate, e is expert
difficulty = 'b'
for x in range (1, 618):
    for y in range (1, 800):
        #We only used this restriction for the beginner graphs
        #in order to get beginner paths at relevant positions
        if(elevation[x][y] > 1800):
            #For each direction, calculate slope and determine whether
            #that point can be descended to
            #Turn pixel white if it is going to another pixel, and turn
            #the pixel being descend to black
            #If pixel is still white by the end, then that means
            #that no other pixel has descended to it and it is
            #included in a path
            slope = (elevation[x][y] - elevation[x][y+1])/26
            if(minslope < slope < maxslope):
                graph[x][y][0] = 1
                if(pix[x,y] != (0,0,0)):
                    pix[x,y] = (255,255,255)
                pix[x,y+1] = (0,0,0)
            slope = (elevation[x][y] - elevation[x+1][y])/26
            if(minslope < slope < maxslope):
                graph[x][y][1] = 1
                if(pix[x,y] != (0,0,0)):
                    pix[x,y] = (255,255,255)
                pix[x+1,y] = (0,0,0)
            slope = (elevation[x][y] - elevation[x][y-1])/26
            if(minslope < slope < maxslope):
                graph[x][y][2] = 1
                if(pix[x,y] != (0,0,0)):
                    pix[x,y] = (255,255,255)
                pix[x,y-1] = (0,0,0)
            slope = (elevation[x][y] - elevation[x-1][y])/26
            if(minslope < slope < maxslope):
                graph[x][y][3] = 1
                if(pix[x,y] != (0,0,0)):
                    pix[x,y] = (255,255,255)
                pix[x-1,y] = (0,0,0)

#Find start values by finding pixels that are still white
starts = []
for x in range (1,618):
    for y in range(1,800):
        if(elevation[x][y] != 0):
            if(pix[x,y] == (255,255,255)):
               starts.append([x,y])

#Save
img.show()
img.save("images/descentgraph" + difficulty + ".png")
with open("descentgraph" + difficulty + ".txt", "wb") as f:
    pickle.dump(graph, f)

with open("starts" + difficulty + ".txt","wb") as fb:
    pickle.dump(starts, fb)

#Find end values by finding points where pixel can no longer descend
ends = []
```

```
for x in range (1,618):
    for y in range(1, 800):
        if (elevation[x][y] != 0):
            if (graph[x][y][0] == 0 and graph[x][y][1] == 0 \
                and graph[x][y][2] == 0 and graph[x][y][3] == 0 \
                and pix[x,y] == (0,0,0)):
                ends.append([x,y])

with open("ends" + difficulty + ".txt","wb") as fba:
    pickle.dump(ends, fba)
```

8.5 Longest Paths

```
import PIL
import PIL.Image
import io
import json
import urllib.request as urllib
import math
import pickle
import numpy as np


#e for expert, i for intermediate, b for beginner
difficulty = 'b'

def intersect(p1, p2):
    '''
    do p1 and p2 intersect?
    '''
    status = False
    for i in range(1,len(p1)-1):
        a = p1[i]

        if a in p2:
            status = True
            break
    return (status)

def invicinity(p1, p2):
    '''
    do two paths start too close to eachother?
    '''
    status= False
    if(0 < (math.fabs(p1[0][0] - p2[0][0])+math.fabs(p1[0][1] - p2[0][1])) < 10)
```

```
        status = True
    return (status)

def clone(array):
    #Copy array
    new_array = []
    for elem in array:
        new_array.append(elem)
    return new_array

def travel(point, path):
    #Algorithm to facilitate Depth First Search using recursion
    global paths
    x = point[0]
    y = point[1]
    path.append([x,y])
    #For each direction, if the point can descend into another point
    #then do so and calculate the path from that point
    if(graph[x][y][0] == 1):
        paths.append(travel([x,y+1],path))
        path = []
    if(graph[x][y][1] == 1):
        paths.append(travel([x+1,y],path))
        path = []
    if(graph[x][y][2] == 1):
        paths.append(travel([x,y-1],path))
        path = []
    if(graph[x][y][3] == 1):
        paths.append(travel([x-1,y],path))
        path = []
    return path

#Load elevations
with open("elevationlist2.txt", "rb") as fp:
    elevation = pickle.load(fp)

#Load descentgraph from the DescentGraph program
with open("descentgraph" + difficulty + ".txt", "rb") as f:
    graph = pickle.load(f)

#Load the start values from the DescentGraph program
with open("starts" + difficulty + ".txt","rb") as fb:
    starts = pickle.load(fb)


img = PIL.Image.open("images/topogradient2.png")
img = img.convert('RGB')
w, h = img.size

pix = img.load()

toppaths = []
totallength = 0
longestrun = 0
longestpath = []
#Calculate the path from each start point
for point in starts:
    paths = []
    path = []
    travel(point,path)
    for path1 in paths:
        #If path is at least 20 pixels long, then store it
        if(len(path1) > 20):
            toppaths.append(path1)
            print(path1)

#Calculate the total length of the paths and render them
#in a visual, where white is a starting point
#and black is the path
for path1 in toppaths:
    for point in path1:
        if(path1.index(point) == 0):
            pix[point[0],point[1]] = (255,255,255)
        else:
            pix[point[0],point[1]] = (0,0,0)
    if(len(path1) > longestrun):
        longestrun=len(path1)
        longestpath = path1
    totallength = totallength + len(path1)

#Save
img.show()
img.save("images/toppaths" + difficulty + ".png")
print(longestrun)
print(totallength)

with open("toppaths" + difficulty + ".txt","wb") as fba:
    pickle.dump(toppaths, fba)
```

8.6 Genetic Algorithm

```
import PIL
import PIL.Image
import io
import json
import urllib.request as urllib
import math
import pickle
import numpy as np
import random

'''
A genetic algorithm that creates the best ski scheme using
data and functions previously generated
'''

#Load elevations
with open("elevationlist2.txt","rb") as fp:
    elevation = pickle.load(fp)

#Load beginner paths
with open("toppathsb.txt","rb") as f1:
    topb = pickle.load(f1)

#Load expert paths
with open("toppathse.txt","rb") as f2:
    tope = pickle.load(f2)

#Load intermediate paths
with open("toppathsi.txt","rb") as f3:
    topi = pickle.load(f3)

img = PIL.Image.open("images/topogradient2.png")
img = img.convert('RGB')
w, h = img.size
pix = img.load()

import numpy
import pickle
import PIL
import PIL.Image
with open("elevationlist2.txt", "rb") as fp:
    elevation = pickle.load(fp)
```

```
def weighpath(path, difficulty):
    '''
    how good is a path?
    output is a value according to a z-score
    '''
    hdist = len(path)
    vdist = (elevation[path[0][0]][path[0][1]] \
            -elevation[path[hdist -1][0]][path[hdist -1][1]])/26

    diffx = path[hdist -1][0] - path[0][0]
    diffy = path[hdist -1][1] - path[1][1]
    diff = math.floor((diffx**2 + diffy**2)**0.5)
    #calculate a curve index by dividing distace traveled
    #by the distance of the direct path
    #curve = 1 when the path is a line
    curve = hdist/diff

    lenpath = len(path)
    neslope = (path[lenpath -1][0] - path[0][0] - path[lenpath -1][1] + path[0][1]
    #calculates how "northeastern" the path is
    #it's delta y (in north direction) + delta x (in east direction) divided by

    opticount=0
    for i in range(0,len(path)-1):
        status = True
        startelevation = elevation[path[i][0]][path[i][1]]
        endelevation = elevation[path[i+1][0]][path[i+1][1]]
        for i in (-1,0,1):
            for j in (-1,0,1):
                if elevation[path[i][0]+i][path[i][1]+j] > endelevation:
                    status = False
        if status:
            opticount+=1

    contour = opticount/diff
    '''
    we want to base the weight off of
    hdist (horizontal distance)
    vdist (vertical distance)
    curve
    neslope
    contour
    '''
    if(difficulty == 'e'):
        return(10+.4*(hdist -38.46)/3.56372478273 + .15*(curve -1.3665741441)\
                /0.175940150257 + .15*(neslope)/0.705382132273)
    if(difficulty == 'i'):
        return(10+.4*(hdist-38.46)/4.440277795 + .15*(curve-1.36810281238)\
        /0.188561533492 + .15*(neslope)/0.350936016599)
    if(difficulty == 'b'):
        return(10+.4*(hdist-38.46)/3.2076488257 + .15*(curve-1.34708156813)\
        /0.231446573523 + .15*(neslope)/0.429224907032)


def intersect(p1, p2):
    '''
    do p1 and p2 intersect?
    '''
    status = False
    for i in range(1,len(p1)-1):
        a = p1[i]
        if a in p2:
            status = True

            break
    return (status)

def invicinity(p1, p2):
    '''
    do two paths start too close to eachother?
    '''
    status= False
    if(0 < (math.fabs(p1[0][0] - p2[0][0])+math.fabs(p1[0][1] - p2[0][1])) < 10)
        status = True
    return (status)

def fitness(paths, difficulty):
    #calculates the total fitness for a set of paths
    rating = 0
    for path in paths:
        rating = rating + weighpath(path, difficulty)
    return rating

def mutate(paths, pathlist):
    #mutate a set of paths
    new_paths = clone(paths)
    random_paths = get_random_paths(20, pathlist)
    for i in range (len(random_paths)):
        #First makes sure that the paths being added do
        #not intersect the previously generated paths
        #in the genetic algorithm
        works = True
        for apath in fixedpaths:
            if(intersect(random_paths[i],apath)):
                works = False
        if(works):
            flag = True
            for k in range (len(paths)):
                #If new path does not intersect a path in the old generation
                #then add it
                #Otherwise, remove the path in the old generation and add
                #the new path
                if(intersect(random_paths[i],new_paths[k]) \
                   or invicinity(random_paths[i],new_paths[k])):
                    del new_paths[k]
                    new_paths.append(random_paths[i])
                    flag = False
                    break
            if(flag):
                new_paths.append(random_paths[i])

    return(new_paths)

def clone(ary):
    #Function to clone an array
    ary2 = []
    for i in ary:
        ary2.append(i)
    return ary2

def get_random_paths(x, pathlist):
    #Randomly generates paths from the list of paths of
    #a certain difficulty
    random_paths = []
    for i in range(x):
        path = pathlist[random.randint(0, len(pathlist)-1)]
        random_paths.append(path)
    return random_paths

def customsort(pathss, difficulty):
    #Sorts the generation based on fitness
    new_pathss = []
    fitnesslist = []
    for i in range(0, len(pathss)):
        fitnesslist.append(fitness(pathss[i], difficulty))
    for k in range(0, len(pathss)):
        found = False
        for i in range (0, len(new_pathss)):
            if((fitnesslist[k] < fitnesslist[i]) and not found):
                new_pathss.insert(i+1, pathss[k])
                found = True
        if(not found):
            new_pathss.insert(0,pathss[k])

    return new_pathss


POP = 20 #population size
ITER = 250 #number of iterations
nstart = 0 #number of paths to start with
#nstart starts at 0 because our algorithm will
#continuously adding paths. Starting it at 0
#will not change much except for minimizing possible
#intersections when initiating the generation

difficulty = 'e' #start with the expert paths

fixedpaths = [] #set of slopes that will not be changed

#Add the expert slopes
gen = []
#Initiate the generation
for i in range(POP):
    gen.append(get_random_paths(nstart, tope))
for i in range(ITER):
    gen = customsort(gen, difficulty)
    print("Generation:", i, "Best:", fitness(gen[0],difficulty))
    #mutate all paths except for the first two
    for k in range(2, POP-1):
        gen[k] = mutate(gen[k%2], tope)

print("FINAL BEST:", len(gen[0]))
totallength = 0
#Find total length of slopes
for path in gen[0]:
    totallength += len(path)
print(totallength)

#Fix the paths and render them into a visual
for path1 in gen[0]:
    fixedpaths.append(path1)
    for point in path1:
        if(path1.index(point) == 0):
            pix[point[0],point[1]] = (255,255,255)
        else:
            pix[point[0],point[1]] = (0,0,0)

img.show()

#Add the intermediate slopes
gen = []
difficulty = 'i'
#Initiate the generation
for i in range(POP):
    gen.append(get_random_paths(nstart, topi))

#Start the Genetic Algorithm
for i in range(ITER):
    gen = customsort(gen, difficulty)
    print("Generation:", i
          , "Best:", fitness(gen[0],difficulty))
    #mutate all paths except for the first two
    for k in range(2, POP-1):
        gen[k] = mutate(gen[k%2], topi)

#Save expert slopes
with open("feslopes.txt","wb") as file:
    pickle.dump(gen, file)

print("FINAL BEST:", len(gen[0]))
#Find total length of slopes
for path in gen[0]:
    totallength += len(path)
print(totallength)

#Fix paths and render them into a visual
for path1 in gen[0]:
    fixedpaths.append(path1)
    for point in path1:
        if(path1.index(point) == 0):
            pix[point[0],point[1]] = (255,255,255)
        else:
            pix[point[0],point[1]] = (0,0,255)

img.show()
#Add the beginner slopes
gen = []
difficulty = 'b'
#Initiate the generation
for i in range(POP):
    gen.append(get_random_paths(nstart, topb))

#Start the GenAlg
for i in range(ITER):
    gen = customsort(gen, difficulty)
    print("Generation:", i
          , "Best:", fitness(gen[0],difficulty))
    #mutate all paths except for the first two
    for k in range(2, POP-1):
        gen[k] = mutate(gen[k%2], topb)

#Save intermediate slopes
with open("fislopes.txt","wb") as file:
    pickle.dump(gen, file)

print("FINAL BEST:", len(gen[0]))
for path in gen[0]:
    totallength += len(path)
print(totallength)

#Fix the paths and render them into a visual
for path1 in gen[0]:
    fixedpaths.append(path1)
    for point in path1:
        if(path1.index(point) == 0):
            pix[point[0],point[1]] = (255,255,255)
        else:
            pix[point[0],point[1]] = (0,255,0)

#Save beginner slopes
with open("fbslopes.txt","wb") as file:
    pickle.dump(gen, file)

#Save the files
with open("finalslopepaths.txt","wb") as file:
    pickle.dump(fixedpaths, file)

img.show()
img.save("images/final.png")
```

9 Appendix B

The optimal ski scheme represented by a list of paths, with each path represented by a list of pixels that it is composed of. [282, 392][283, 392][284, 392][285, 392][286, 392][287, 392][288, 392][289, 392][290, 392][291, 392][292, 392][293, 392][294, 392][295, 392][296, 392][297, 392][298, 392][299, 392][300,

392][301, 392][302, 392][303, 392][304, 392] [206, 239][206, 238][207, 238][208,
238][209, 238][210, 238][211, 238][212, 238][213, 238][214, 238][215, 238][216,
238][217, 238][218, 238][219, 238][220, 238][221, 238][222, 238][222, 237][222,
236][223, 236][224, 236] [243, 308][244, 308][245, 308][246, 308][247, 308][248,
308][249, 308][250, 308][251, 308][252, 308][253, 308][254, 308][255, 308][256,
308][257, 308][258, 308][259, 308][260, 308][261, 308][262, 308][263, 308][264,
308][265, 308][266, 308][267, 308][268, 308][269, 308][270, 308][271, 308][272,
308] [266, 322][267, 322][268, 322][269, 322][270, 322][271, 322][272, 322][273,
322][274, 322][275, 322][276, 322][277, 322][278, 322][278, 323][278, 324][279,
324][280, 324][281, 324][282, 324][283, 324][284, 324][285, 324][286, 324][287,
324][288, 324] [332, 692][333, 692][334, 692][335, 692][336, 692][337, 692][338,
692][339, 692][340, 692][341, 692][342, 692][343, 692][344, 692][345, 692][346,
692][347, 692][348, 692][349, 692][350, 692][351, 692][352, 692] [262, 338][263,
338][264, 338][265, 338][266, 338][267, 338][268, 338][269, 338][270, 338][271,
338][272, 338][273, 338][274, 338][274, 339][274, 340][274, 341][274, 342][275,
342][276, 342][277, 342][278, 342][279, 342][280, 342][280, 343][280, 344][281,
344][282, 344] [290, 355][290, 354][291, 354][292, 354][293, 354][294, 354][295,
354][296, 354][297, 354][298, 354][299, 354][300, 354][301, 354][302, 354][303,
354][304, 354][305, 354][306, 354][307, 354][308, 354][309, 354][310, 354] [266,
322][267, 322][268, 322][269, 322][270, 322][271, 322][272, 322][273, 322][274,
322][275, 322][276, 322][277, 322][278, 322][278, 323][278, 324][279, 324][280,
324][281, 324][282, 324][283, 324][284, 324][285, 324][286, 324][287, 324][288,
324] [227, 260][228, 260][229, 260][230, 260][231, 260][232, 260][233, 260][234,
260][235, 260][236, 260][237, 260][238, 260][239, 260][240, 260][241, 260][242,
260][242, 259][242, 258][243, 258][244, 258][244, 257][244, 256][245, 256][246,
256][246, 255][246, 254] [277, 626][278, 626][279, 626][280, 626][281, 626][282,
626][283, 626][284, 626][285, 626][286, 626][286, 627][286, 628][287, 628][288,
628][289, 628][290, 628][291, 628][292, 628][293, 628][294, 628][295, 628][296,
628][297, 628][298, 628][299, 628][300, 628] [244, 298][245, 298][246, 298][247,
298][248, 298][249, 298][250, 298][251, 298][252, 298][253, 298][254, 298][255,
298][256, 298][257, 298][258, 298][259, 298][260, 298][261, 298][262, 298][263,
298][264, 298][265, 298][266, 298][266, 299][266, 300][266, 301][266, 302][267,
302][268, 302] [322, 676][323, 676][324, 676][325, 676][326, 676][326, 675][326,
674][326, 673][326, 672][326, 671][326, 670][327, 670][328, 670][328, 669][328,
668][329, 668][330, 668][331, 668][332, 668][333, 668][334, 668][334, 667][334,
666] [242, 452][242, 453][242, 454][242, 455][242, 456][242, 457][242, 458][242,
459][242, 460][242, 461][242, 462][243, 462][244, 462][245, 462][246, 462][247,
462][248, 462][249, 462][250, 462][251, 462][252, 462] [210, 194][210, 193][210,
192][210, 191][210, 190][210, 189][210, 188][209, 188][208, 188][207, 188][206,
188][206, 187][206, 186][205, 186][204, 186][204, 185][204, 184][203, 184][202,
184][202, 183][202, 182] [266, 322][267, 322][268, 322][269, 322][270, 322][271,
322][272, 322][273, 322][274, 322][275, 322][276, 322][277, 322][278, 322][278,
323][278, 324][279, 324][280, 324][281, 324][282, 324][283, 324][284, 324][285,
324][286, 324][287, 324][288, 324] [258, 588][259, 588][260, 588][260, 587][260,
586][261, 586][262, 586][263, 586][264, 586][265, 586][266, 586][267, 586][268,
586][269, 586][270, 586][271, 586][272, 586][272, 585][272, 584][272, 583][272,

582] [258, 572][258, 573][258, 574][259, 574][260, 574][261, 574][262, 574][263, 574][264, 574][265, 574][266, 574][267, 574][268, 574][269, 574][270, 574][271, 574][272, 574][273, 574][274, 574][275, 574][276, 574][277, 574][278, 574][279, 574][280, 574] [270, 598][271, 598][272, 598][273, 598][274, 598][275, 598][276, 598][277, 598][278, 598][278, 599][278, 600][278, 601][278, 602][278, 603][278, 604][278, 605][278, 606][279, 606][280, 606][281, 606][282, 606] [242, 306][243, 306][244, 306][245, 306][246, 306][247, 306][248, 306][249, 306][250, 306][251, 306][252, 306][253, 306][254, 306][255, 306][256, 306][257, 306][258, 306][259, 306][260, 306][261, 306][262, 306][262, 307][262, 308][263, 308][264, 308][265, 308][266, 308][267, 308][268, 308][269, 308][270, 308][271, 308][272, 308] [265, 594][266, 594][267, 594][268, 594][269, 594][270, 594][271, 594][272, 594][273, 594][274, 594][275, 594][276, 594][277, 594][278, 594][278, 595][278, 596][278, 597][278, 598][278, 599][278, 600][278, 601][278, 602][278, 603][278, 604][278, 605][278, 606][279, 606][280, 606][281, 606][282, 606] [203, 280][204, 280][205, 280][206, 280][207, 280][208, 280][209, 280][210, 280][211, 280][212, 280][213, 280][214, 280][215, 280][216, 280][217, 280][218, 280][219, 280][220, 280][221, 280][222, 280][223, 280][224, 280][225, 280][226, 280][227, 280][228, 280][229, 280][230, 280][231, 280][232, 280] [253, 374][254, 374][255, 374][256, 374][257, 374][258, 374][259, 374][260, 374][261, 374][262, 374][263, 374][264, 374][265, 374][266, 374][267, 374][268, 374][269, 374][270, 374][271, 374][272, 374][273, 374][274, 374][275, 374][276, 374][277, 374][278, 374][279, 374][280, 374][281, 374][282, 374][283, 374][284, 374][285, 374][286, 374] [190, 296][191, 296][192, 296][192, 297][192, 298][192, 299][192, 300][192, 301][192, 302][192, 303][192, 304][193, 304][194, 304][195, 304][196, 304][197, 304][198, 304][199, 304][200, 304][201, 304][202, 304][203, 304][204, 304][205, 304][206, 304][207, 304][208, 304] [228, 242][228, 241][228, 240][228, 239][228, 238][229, 238][230, 238][230, 237][230, 236][231, 236][232, 236][232, 235][232, 234][233, 234][234, 234][234, 233][234, 232][234, 231][234, 230][235, 230][236, 230][237, 230][238, 230][239, 230][240, 230][241, 230][242, 230] [284, 619][284, 618][285, 618][286, 618][287, 618][288, 618][289, 618][290, 618][291, 618][292, 618][293, 618][294, 618][295, 618][296, 618][297, 618][298, 618][299, 618][300, 618][301, 618][302, 618][303, 618][304, 618][305, 618][306, 618] [244, 374][245, 374][246, 374][247, 374][248, 374][248, 375][248, 376][249, 376][250, 376][251, 376][252, 376][253, 376][254, 376][255, 376][256, 376][257, 376][258, 376][259, 376][260, 376][261, 376][262, 376][263, 376][264, 376][265, 376][266, 376][267, 376][268, 376][269, 376][270, 376][271, 376][272, 376] [206, 139][206, 138][206, 137][206, 136][207, 136][208, 136][209, 136][210, 136][210, 135][210, 134][210, 133][210, 132][210, 131][210, 130][211, 130][212, 130][212, 129][212, 128][212, 127][212, 126][212, 125][212, 124][212, 123][212, 122][212, 121][212, 120] [195, 302][196, 302][197, 302][198, 302][199, 302][200, 302][201, 302][202, 302][203, 302][204, 302][205, 302][206, 302][207, 302][208, 302][209, 302][210, 302][211, 302][212, 302][213, 302][214, 302][215, 302][216, 302][217, 302][218, 302][219, 302][220, 302][221, 302][222, 302][223, 302][224, 302] [243, 308][244, 308][245, 308][246, 308][247, 308][248, 308][249, 308][250, 308][251, 308][252, 308][253, 308][254, 308][255, 308][256, 308][257, 308][258, 308][259, 308][260, 308][261, 308][262, 308][263, 308][264, 308][265, 308][266, 308][267, 308][268, 308][269, 308][270, 308][271, 308][272,

```
308] [271, 294][272, 294][273, 294][274, 294][275, 294][276, 294][277, 294][278,
294][279, 294][280, 294][281, 294][282, 294][283, 294][284, 294][285, 294][286,
294][287, 294][288, 294][289, 294][290, 294][291, 294][292, 294][293, 294][294,
294][295, 294][296, 294] [242, 474][242, 473][242, 472][242, 471][242, 470][242,
469][242, 468][243, 468][244, 468][245, 468][246, 468][247, 468][248, 468][249,
468][250, 468][251, 468][252, 468][253, 468][254, 468][255, 468][256, 468][257,
468][258, 468][259, 468][260, 468][261, 468][262, 468][263, 468][264, 468] [306,
645][306, 644][307, 644][308, 644][309, 644][310, 644][311, 644][312, 644][313,
644][314, 644][315, 644][316, 644][317, 644][318, 644][319, 644][320, 644][321,
644][322, 644][323, 644][324, 644][325, 644][326, 644][327, 644][328, 644][329,
644][330, 644] [232, 241][232, 240][233, 240][234, 240][235, 240][236, 240][237,
240][238, 240][239, 240][240, 240][240, 241][240, 242][240, 243][240, 244][240,
245][240, 246][241, 246][242, 246][243, 246][244, 246][245, 246][246, 246][247,
246][248, 246] [209, 150][208, 150][208, 149][208, 148][208, 147][208, 146][209,
146][210, 146][210, 145][210, 144][211, 144][212, 144][212, 143][212, 142][212,
141][212, 140][212, 139][212, 138][212, 137][212, 136][212, 135][212, 134][212,
133][212, 132][212, 131][212, 130][212, 129][212, 128][212, 127][212, 126][212,
125][212, 124][212, 123][212, 122][212, 121][212, 120] [209, 150][208, 150][208,
149][208, 148][208, 147][208, 146][209, 146][210, 146][210, 145][210, 144][211,
144][212, 144][212, 143][212, 142][212, 141][212, 140][212, 139][212, 138][212,
137][212, 136][212, 135][212, 134][212, 133][212, 132][212, 131][212, 130][212,
129][212, 128][212, 127][212, 126][212, 125][212, 124][212, 123][212, 122][212,
121][212, 120] [293, 588][294, 588][295, 588][296, 588][297, 588][298, 588][299,
588][300, 588][301, 588][302, 588][303, 588][304, 588][305, 588][306, 588][307,
588][308, 588][309, 588][310, 588][311, 588][312, 588][313, 588][314, 588][315,
588][316, 588][317, 588][318, 588][319, 588][320, 588] [258, 443][258, 442][259,
442][260, 442][261, 442][262, 442][262, 441][262, 440][263, 440][264, 440][265,
440][266, 440][267, 440][268, 440][269, 440][270, 440][271, 440][272, 440][272,
439][272, 438][272, 437][272, 436] [246, 372][246, 373][246, 374][247, 374][248,
374][248, 375][248, 376][249, 376][250, 376][251, 376][252, 376][253, 376][254,
376][255, 376][256, 376][257, 376][258, 376][259, 376][260, 376][261, 376][262,
376][263, 376][264, 376][265, 376][266, 376][267, 376][268, 376][269, 376][270,
376][271, 376][272, 376] [260, 432][261, 432][262, 432][263, 432][264, 432][265,
432][266, 432][267, 432][268, 432][269, 432][270, 432][271, 432][272, 432][273,
432][274, 432][275, 432][276, 432][277, 432][278, 432][279, 432][280, 432][281,
432][282, 432] [216, 186][216, 185][216, 184][215, 184][214, 184][213, 184][212,
184][212, 183][212, 182][212, 181][212, 180][212, 179][212, 178][211, 178][210,
178][210, 177][210, 176][209, 176][208, 176][207, 176][206, 176][205, 176][204,
176][203, 176][202, 176][201, 176][200, 176][199, 176][198, 176][198, 175][198,
174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [218, 169][218,
168][219, 168][220, 168][221, 168][222, 168][222, 167][222, 166][223, 166][224,
166][224, 165][224, 164][225, 164][226, 164][227, 164][228, 164][229, 164][230,
164][231, 164][232, 164][232, 163][232, 162][232, 161][232, 160][232, 159][232,
158] [214, 188][214, 187][214, 186][214, 185][214, 184][213, 184][212, 184][212,
183][212, 182][212, 181][212, 180][212, 179][212, 178][211, 178][210, 178][210,
177][210, 176][209, 176][208, 176][207, 176][206, 176][205, 176][204, 176][203,
```

176][202, 176][201, 176][200, 176][199, 176][198, 176][198, 175][198, 174][197,
174][196, 174][196, 173][196, 172][195, 172][194, 172] [205, 182][204, 182][204,
181][204, 180][203, 180][202, 180][202, 179][202, 178][201, 178][200, 178][200,
177][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196,
173][196, 172][195, 172][194, 172] [218, 169][218, 168][219, 168][220, 168][221,
168][222, 168][222, 167][222, 166][223, 166][224, 166][224, 165][224, 164][225,
164][226, 164][227, 164][228, 164][229, 164][230, 164][231, 164][232, 164][232,
163][232, 162][232, 161][232, 160][232, 159][232, 158] [253, 374][254, 374][255,
374][256, 374][257, 374][258, 374][259, 374][260, 374][261, 374][262, 374][263,
374][264, 374][265, 374][266, 374][267, 374][268, 374][269, 374][270, 374][271,
374][272, 374][273, 374][274, 374][275, 374][276, 374][277, 374][278, 374][279,
374][280, 374][281, 374][282, 374][283, 374][284, 374][285, 374][286, 374] [196,
287][196, 286][197, 286][198, 286][199, 286][200, 286][201, 286][202, 286][203,
286][204, 286][205, 286][206, 286][207, 286][208, 286][209, 286][210, 286][211,
286][212, 286][213, 286][214, 286][215, 286][216, 286][217, 286][218, 286][219,
286][220, 286][221, 286][222, 286][223, 286][224, 286] [214, 188][214, 187][214,
186][214, 185][214, 184][213, 184][212, 184][212, 183][212, 182][212, 181][212,
180][212, 179][212, 178][211, 178][210, 178][210, 177][210, 176][209, 176][208,
176][207, 176][206, 176][205, 176][204, 176][203, 176][202, 176][201, 176][200,
176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196,
172][195, 172][194, 172] [222, 169][222, 168][222, 167][222, 166][223, 166][224,
166][224, 165][224, 164][225, 164][226, 164][227, 164][228, 164][229, 164][230,
164][231, 164][232, 164][232, 163][232, 162][232, 161][232, 160][232, 159][232,
158] [208, 141][208, 140][209, 140][210, 140][210, 139][210, 138][210, 137][210,
136][210, 135][210, 134][210, 133][210, 132][210, 131][210, 130][211, 130][212,
130][212, 129][212, 128][212, 127][212, 126][212, 125][212, 124][212, 123][212,
122][212, 121][212, 120] [197, 290][198, 290][198, 291][198, 292][199, 292][200,
292][201, 292][202, 292][203, 292][204, 292][205, 292][206, 292][207, 292][208,
292][209, 292][210, 292][211, 292][212, 292][213, 292][214, 292][215, 292][216,
292][217, 292][218, 292][219, 292][220, 292][221, 292][222, 292][223, 292][224,
292] [222, 169][222, 168][222, 167][222, 166][223, 166][224, 166][224, 165][224,
164][225, 164][226, 164][227, 164][228, 164][229, 164][230, 164][231, 164][232,
164][232, 163][232, 162][232, 161][232, 160][232, 159][232, 158] [200, 282][201,
282][202, 282][203, 282][204, 282][205, 282][206, 282][207, 282][208, 282][209,
282][210, 282][211, 282][212, 282][213, 282][214, 282][215, 282][216, 282][217,
282][218, 282][219, 282][220, 282][221, 282][222, 282][223, 282][224, 282][225,
282][226, 282][227, 282][228, 282][229, 282][230, 282] [211, 180][210, 180][210,
179][210, 178][210, 177][210, 176][209, 176][208, 176][207, 176][206, 176][205,
176][204, 176][203, 176][202, 176][201, 176][200, 176][199, 176][198, 176][198,
175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [214,
174][214, 173][214, 172][214, 171][214, 170][214, 169][214, 168][214, 167][214,
166][214, 165][214, 164][215, 164][216, 164][216, 163][216, 162][216, 161][216,
160][217, 160][218, 160][218, 159][218, 158][218, 157][218, 156][219, 156][220,
156] [216, 186][216, 185][216, 184][215, 184][214, 184][213, 184][212, 184][212,
183][212, 182][212, 181][212, 180][212, 179][212, 178][211, 178][210, 178][210,
177][210, 176][209, 176][208, 176][207, 176][206, 176][205, 176][204, 176][203,

176][202, 176][201, 176][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [212, 151][212, 150][212, 149][212, 148][212, 147][212, 146][212, 145][212, 144][212, 143][212, 142][212, 141][212, 140][212, 139][212, 138][212, 137][212, 136][212, 135][212, 134][212, 133][212, 132][212, 131][212, 130][212, 129][212, 128][212, 127][212, 126][212, 125][212, 124][212, 123][212, 122][212, 121][212, 120] [256, 499][256, 498][257, 498][258, 498][259, 498][260, 498][261, 498][262, 498][263, 498][264, 498][265, 498][266, 498][267, 498][268, 498][269, 498][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [256, 503][256, 502][257, 502][258, 502][259, 502][260, 502][261, 502][262, 502][263, 502][264, 502][265, 502][266, 502][267, 502][268, 502][269, 502][270, 502][270, 501][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [260, 501][260, 500][261, 500][262, 500][263, 500][264, 500][265, 500][266, 500][267, 500][268, 500][269, 500][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [216, 188][216, 189][216, 190][216, 191][216, 192][217, 192][218, 192][219, 192][220, 192][220, 193][220, 194][221, 194][222, 194][223, 194][224, 194][225, 194][226, 194][227, 194][228, 194][229, 194][230, 194] [279, 628][280, 628][281, 628][282, 628][283, 628][284, 628][285, 628][286, 628][287, 628][288, 628][289, 628][290, 628][291, 628][292, 628][293, 628][294, 628][295, 628][296, 628][297, 628][298, 628][299, 628][300, 628] [197, 288][198, 288][198, 289][198, 290][198, 291][198, 292][199, 292][200, 292][201, 292][202, 292][203, 292][204, 292][205, 292][206, 292][207, 292][208, 292][209, 292][210, 292][211, 292][212, 292][213, 292][214, 292][215, 292][216, 292][217, 292][218, 292][219, 292][220, 292][221, 292][222, 292][223, 292][224, 292] [222, 169][222, 168][222, 167][222, 166][223, 166][224, 166][224, 165][224, 164][225, 164][226, 164][227, 164][228, 164][229, 164][230, 164][231, 164][232, 164][232, 163][232, 162][232, 161][232, 160][232, 159][232, 158] [203, 162][202, 162][202, 161][202, 160][202, 159][202, 158][202, 157][202, 156][202, 155][202, 154][201, 154][200, 154][199, 154][198, 154][197, 154][196, 154][196, 153][196, 152][195, 152][194, 152][193, 152][192, 152][191, 152][190, 152] [254, 499][254, 498][255, 498][256, 498][257, 498][258, 498][259, 498][260, 498][261, 498][262, 498][263, 498][264, 498][265, 498][266, 498][267, 498][268, 498][269, 498][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [220, 169][220, 168][221, 168][222, 168][222, 167][222, 166][223, 166][224, 166][224, 165][224, 164][225, 164][226, 164][227, 164][228, 164][229, 164][230, 164][231, 164][232, 164][232, 163][232, 162][232, 161][232, 160][232, 159][232, 158] [256, 501][256, 500][257, 500][258, 500][259, 500][260, 500][261, 500][262, 500][263, 500][264, 500][265, 500][266, 500][267, 500][268, 500][269, 500][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [205, 182][204, 182][204, 181][204, 180][203, 180][202, 180][202, 179][202, 178][201, 178][200, 178][200, 177][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [310, 645][310, 644][311, 644][312, 644][313, 644][314, 644][315, 644][316, 644][317, 644][318, 644][319, 644][320, 644][321, 644][322, 644][323, 644][324, 644][325, 644][326, 644][327, 644][328, 644][329, 644][330, 644] [248, 235][248, 234][249, 234][250, 234][251, 234][252, 234][253, 234][254, 234][254, 233][254, 232][255, 232][256, 232][256, 231][256, 230][257, 230][258, 230][259, 230][260, 230][261, 230][262, 230][263, 230][264, 230][265, 230][266, 230] [210, 166][210, 167][210, 168][209, 168][208, 168][207, 168][206, 168][205, 168][204, 168][204, 169][204, 170][203, 170][202, 170][201, 170][200, 170][199, 170][198, 170][197, 170][196, 170][195, 170][194, 170][193, 170][192, 170][192, 169][192, 168][192, 167][192, 166] [206, 265][206, 264][206, 263][206, 262][206, 261][206, 260][206, 259][206, 258][206, 257][206, 256][206, 255][206, 254][207, 254][208, 254][209, 254][210, 254][211, 254][212, 254][212, 253][212, 252][213, 252][214, 252][215, 252][216, 252][216, 251][216, 250] [215, 176][214, 176][213, 176][212, 176][211, 176][210, 176][209, 176][208, 176][207, 176][206, 176][205, 176][204, 176][203, 176][202, 176][201, 176][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [197, 292][198, 292][199, 292][200, 292][201, 292][202, 292][203, 292][204, 292][205, 292][206, 292][207, 292][208, 292][209, 292][210, 292][211, 292][212, 292][213, 292][214, 292][215, 292][216, 292][217, 292][218, 292][219, 292][220, 292][221, 292][222, 292][223, 292][224, 292] [182, 189][182, 188][183, 188][184, 188][184, 187][184, 186][185, 186][186, 186][187, 186][188, 186][188, 185][188, 184][188, 183][188, 182][188, 181][188, 180][188, 179][188, 178][189, 178][190, 178][191, 178][192, 178][192, 177][192, 176] [205, 182][204, 182][204, 181][204, 180][203, 180][202, 180][202, 179][202, 178][201, 178][200, 178][200, 177][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [221, 136][220, 136][220, 135][220, 134][220, 133][220, 132][220, 131][220, 130][220, 129][220, 128][220, 127][220, 126][220, 125][220, 124][219, 124][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [258, 503][258, 502][259, 502][260, 502][261, 502][262, 502][263, 502][264, 502][265, 502][266, 502][267, 502][268, 502][269, 502][270, 502][270, 501][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [221, 140][220, 140][220, 139][220, 138][220, 137][220, 136][220, 135][220, 134][220, 133][220, 132][220, 131][220, 130][220, 129][220, 128][220, 127][220, 126][220, 125][220, 124][219, 124][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [302, 646][303, 646][304, 646][305, 646][306, 646][307, 646][308, 646][309, 646][310, 646][311, 646][312, 646][313, 646][314, 646][315, 646][316, 646][317, 646][318, 646][319, 646][320, 646][321, 646][322, 646][323, 646][324, 646][325, 646][326, 646][327, 646][328, 646][329, 646][330, 646] [215, 186][214, 186][214, 185][214, 184][213, 184][212, 184][212, 183][212, 182][212, 181][212, 180][212, 179][212, 178][211, 178][210, 178][210, 177][210, 176][209, 176][208, 176][207, 176][206, 176][205, 176][204, 176][203, 176][202, 176][201, 176][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [316, 660][317, 660][318, 660][319, 660][320, 660][321, 660][322, 660][323, 660][324, 660][325, 660][326, 660][327, 660][328, 660][329, 660][330, 660][331, 660][332, 660][333, 660][334, 660][335, 660][336, 660] [252, 501][252, 500][253, 500][254, 500][255, 500][256, 500][257, 500][258, 500][259, 500][260, 500][261, 500][262, 500][263, 500][264, 500][265, 500][266, 500][267, 500][268, 500][269, 500][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [238, 282][239, 282][240, 282][241, 282][242, 282][243, 282][244, 282][245, 282][246, 282][247, 282][248, 282][249, 282][250, 282][251, 282][252, 282][253, 282][254, 282][255, 282][256, 282][257, 282][258, 282][259, 282][260, 282][261, 282][262, 282][263, 282][264, 282][265, 282][266, 282][267, 282][268, 282][269, 282][270, 282] [219, 136][218, 136][218, 135][218, 134][218, 133][218, 132][218, 131][218, 130][218, 129][218, 128][218, 127][218, 126][218, 125][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [260, 503][260, 502][261, 502][262, 502][263, 502][264, 502][265, 502][266, 502][267, 502][268, 502][269, 502][270, 502][270, 501][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [258, 499][258, 498][259, 498][260, 498][261, 498][262, 498][263, 498][264, 498][265, 498][266, 498][267, 498][268, 498][269, 498][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [223, 142][222, 142][222, 141][222, 140][222, 139][222, 138][222, 137][222, 136][222, 135][222, 134][222, 133][222, 132][222, 131][222, 130][222, 129][222, 128][222, 127][222, 126][221, 126][220, 126][220, 125][220, 124][219, 124][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [207, 182][206, 182][206, 181][206, 180][205, 180][204, 180][203, 180][202, 180][202, 179][202, 178][201, 178][200, 178][200, 177][200, 176][199, 176][198, 176][198, 175][198, 174][197, 174][196, 174][196, 173][196, 172][195, 172][194, 172] [256, 499][256, 498][257, 498][258, 498][259, 498][260, 498][261, 498][262, 498][263, 498][264, 498][265, 498][266, 498][267, 498][268, 498][269, 498][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [254, 499][254, 498][255, 498][256, 498][257, 498][258, 498][259, 498][260, 498][261, 498][262, 498][263, 498][264, 498][265, 498][266, 498][267, 498][268, 498][269, 498][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [260, 503][260, 502][261, 502][262, 502][263, 502][264, 502][265, 502][266, 502][267, 502][268, 502][269, 502][270, 502][270, 501][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [221, 142][220, 142][220, 141][220, 140][220, 139][220, 138][220, 137][220, 136][220, 135][220, 134][220, 133][220, 132][220, 131][220, 130][220, 129][220, 128][220, 127][220, 126][220, 125][220, 124][219, 124][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [256, 501][256, 500][257, 500][258, 500][259, 500][260, 500][261, 500][262, 500][263, 500][264, 500][265, 500][266, 500][267, 500][268, 500][269, 500][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [221, 136][220, 136][220, 135][220, 134][220, 133][220, 132][220, 131][220, 130][220, 129][220, 128][220, 127][220, 126][220, 125][220, 124][219, 124][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [209, 152][208, 152][208, 151][208, 150][208, 149][208, 148][208, 147][208, 146][209, 146][210, 146][210, 145][210, 144][211, 144][212, 144][212, 143][212, 142][212, 141][212, 140][212, 139][212, 138][212, 137][212, 136][212, 135][212, 134][212, 133][212, 132][212, 131][212, 130][212, 129][212, 128][212, 127][212, 126][212, 125][212, 124][212, 123][212, 122][212, 121][212, 120] [250, 503][250, 502][251, 502][252, 502][253, 502][254, 502][255, 502][256, 502][257, 502][258, 502][259, 502][260, 502][261, 502][262, 502][263, 502][264, 502][265, 502][266, 502][267, 502][268, 502][269, 502][270, 502][270, 501][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [221, 140][220, 140][220, 139][220, 138][220, 137][220, 136][220, 135][220, 134][220, 133][220, 132][220, 131][220, 130][220, 129][220, 128][220, 127][220, 126][220, 125][220, 124][219, 124][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [219, 138][218, 138][218, 137][218, 136][218, 135][218, 134][218, 133][218, 132][218, 131][218, 130][218, 129][218, 128][218, 127][218, 126][218, 125][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [260, 501][260, 500][261, 500][262, 500][263, 500][264, 500][265, 500][266, 500][267, 500][268, 500][269, 500][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [252, 260][252, 259][252, 258][252, 257][252, 256][252, 255][252, 254][253, 254][254, 254][254, 253][254, 252][254, 251][254, 250][255, 250][256, 250][257, 250][258, 250][259, 250][260, 250][261, 250][262, 250][263, 250][264, 250][265, 250][266, 250][267, 250][268, 250] [252, 497][252, 496][253, 496][254, 496][255, 496][256, 496][256, 495][256, 494][257, 494][258, 494][258, 493][258, 492][258, 491][258, 490][258, 489][258, 488][258, 487][258, 486][258, 485][258, 484][259, 484][260, 484][260, 483][260, 482] [260, 501][260, 500][261, 500][262, 500][263, 500][264, 500][265, 500][266, 500][267, 500][268, 500][269, 500][270, 500][270, 499][270, 498][270, 497][270, 496][271, 496][272, 496][272, 495][272, 494][272, 493][272, 492] [199, 294][200, 294][201, 294][202, 294][203, 294][204, 294][205, 294][206, 294][207, 294][208, 294][209, 294][210, 294][211, 294][212, 294][213, 294][214, 294][215, 294][216, 294][217, 294][218, 294][219, 294][220, 294][221, 294][222, 294] [219, 136][218, 136][218, 135][218, 134][218, 133][218, 132][218, 131][218, 130][218, 129][218, 128][218, 127][218, 126][218, 125][218, 124][217, 124][216, 124][216, 123][216, 122][216, 121][216, 120][215, 120][214, 120] [228, 242][228, 241][228, 240][228, 239][228, 238][229, 238][230, 238][230, 237][230, 236][231, 236][232, 236][232, 235][232, 234][233, 234][234, 234][234, 233][234, 232][234, 231][234, 230][235, 230][236, 230][237, 230][238, 230][239, 230][240, 230][241, 230][242, 230] [242, 202][243, 202][244, 202][245, 202][246, 202][247, 202][248, 202][249, 202][250, 202][250, 201][250, 200][251, 200][252, 200][252, 199][252, 198][253, 198][254, 198][255, 198][256, 198][257, 198][258, 198][258, 197][258, 196][259, 196][260, 196] [207, 160][206, 160][206, 159][206, 158][206, 157][206, 156][206, 155][206, 154][205, 154][204, 154][203, 154][202, 154][201, 154][200, 154][199, 154][198, 154][197, 154][196, 154][196, 153][196, 152][195, 152][194, 152][193, 152][192, 152][191, 152][190, 152] [205, 158][204, 158][204, 157][204, 156][204, 155][204, 154][203, 154][202, 154][201, 154][200, 154][199, 154][198, 154][197, 154][196, 154][196, 153][196, 152][195, 152][194, 152][193, 152][192, 152][191, 152][190, 152] [261, 224][262, 224][263, 224][264, 224][265, 224][266, 224][267, 224][268, 224][269, 224][270, 224][271, 224][272, 224][273, 224][274, 224][275, 224][276, 224][277, 224][278, 224][279, 224][280, 224][281, 224][282, 224] [302, 418][302, 417][302, 416][303, 416][304, 416][305, 416][306, 416][307, 416][308, 416][309, 416][310, 416][310, 415][310, 414][311, 414][312, 414][312, 413][312, 412][313, 412][314, 412][314, 411][314, 410] [304, 648][305, 648][306, 648][307, 648][308, 648][309, 648][310, 648][311, 648][312, 648][313, 648][314, 648][315, 648][316, 648][317, 648][318, 648][319, 648][320, 648][321, 648][322, 648][323, 648][324, 648][325,

648][326, 648][327, 648][328, 648][329, 648][330, 648][330, 647][330, 646] [274,
288][275, 288][276, 288][277, 288][278, 288][279, 288][280, 288][281, 288][282,
288][283, 288][284, 288][285, 288][286, 288][286, 287][286, 286][287, 286][288,
286][289, 286][290, 286][291, 286][292, 286][293, 286][294, 286][295, 286][296,
286][297, 286][298, 286][299, 286][300, 286][301, 286][302, 286] [260, 324][261,
324][262, 324][263, 324][264, 324][265, 324][266, 324][267, 324][268, 324][269,
324][270, 324][270, 325][270, 326][271, 326][272, 326][273, 326][274, 326][275,
326][276, 326][277, 326][278, 326] [240, 238][241, 238][242, 238][243, 238][244,
238][245, 238][246, 238][247, 238][248, 238][248, 237][248, 236][249, 236][250,
236][251, 236][252, 236][253, 236][254, 236][254, 237][254, 238][254, 239][254,
240] [298, 494][299, 494][300, 494][301, 494][302, 494][303, 494][304, 494][305,
494][306, 494][306, 495][306, 496][306, 497][306, 498][307, 498][308, 498][309,
498][310, 498][311, 498][312, 498][313, 498][314, 498][315, 498][316, 498] [290,
360][291, 360][292, 360][293, 360][294, 360][295, 360][296, 360][297, 360][298,
360][299, 360][300, 360][301, 360][302, 360][303, 360][304, 360][305, 360][306,
360][307, 360][308, 360][308, 359][308, 358][309, 358][310, 358][311, 358][312,
358][313, 358][314, 358][315, 358][316, 358][317, 358][318, 358] [278, 314][279,
314][280, 314][281, 314][282, 314][283, 314][284, 314][284, 315][284, 316][285,
316][286, 316][287, 316][288, 316][289, 316][290, 316][291, 316][292, 316][293,
316][294, 316][295, 316][296, 316] [322, 602][323, 602][324, 602][325, 602][326,
602][327, 602][328, 602][329, 602][330, 602][331, 602][332, 602][333, 602][334,
602][335, 602][336, 602][336, 601][336, 600][336, 599][336, 598][337, 598][338,
598][339, 598][340, 598] [314, 486][315, 486][316, 486][317, 486][318, 486][319,
486][320, 486][320, 485][320, 484][320, 483][320, 482][320, 481][320, 480][321,
480][322, 480][322, 479][322, 478][322, 477][322, 476][322, 475][322, 474] [250,
476][250, 477][250, 478][251, 478][252, 478][253, 478][254, 478][255, 478][256,
478][256, 479][256, 480][257, 480][258, 480][259, 480][260, 480][261, 480][262,
480][263, 480][264, 480][265, 480][266, 480] [279, 594][280, 594][280, 595][280,
596][280, 597][280, 598][280, 599][280, 600][280, 601][280, 602][280, 603][280,
604][281, 604][282, 604][283, 604][284, 604][285, 604][286, 604][287, 604][288,
604][289, 604][290, 604][291, 604][292, 604] [322, 682][322, 681][322, 680][323,
680][324, 680][325, 680][326, 680][327, 680][328, 680][329, 680][330, 680][331,
680][332, 680][332, 679][332, 678][332, 677][332, 676][333, 676][334, 676][335,
676][336, 676][337, 676][338, 676][339, 676][340, 676][340, 675][340, 674][340,
673][340, 672] [324, 679][324, 678][325, 678][326, 678][327, 678][328, 678][329,
678][330, 678][331, 678][332, 678][332, 677][332, 676][333, 676][334, 676][335,
676][336, 676][337, 676][338, 676][339, 676][340, 676][340, 675][340, 674][340,
673][340, 672] [324, 682][325, 682][326, 682][327, 682][328, 682][329, 682][330,
682][331, 682][332, 682][332, 681][332, 680][332, 679][332, 678][332, 677][332,
676][333, 676][334, 676][335, 676][336, 676][337, 676][338, 676][339, 676][340,
676][340, 675][340, 674][340, 673][340, 672] [324, 682][325, 682][326, 682][327,
682][328, 682][329, 682][330, 682][331, 682][332, 682][332, 681][332, 680][332,
679][332, 678][332, 677][332, 676][333, 676][334, 676][335, 676][336, 676][337,
676][338, 676][339, 676][340, 676][340, 675][340, 674][340, 673][340, 672] [278,
624][279, 624][280, 624][280, 623][280, 622][281, 622][282, 622][282, 621][282,
620][283, 620][284, 620][285, 620][286, 620][287, 620][288, 620][289, 620][290,

620][291, 620][292, 620][293, 620][294, 620][295, 620][296, 620][296, 621][296, 622][297, 622][298, 622][299, 622][300, 622][301, 622][302, 622][303, 622][304, 622][304, 621][304, 620] [236, 280][237, 280][238, 280][239, 280][240, 280][241, 280][242, 280][243, 280][244, 280][245, 280][246, 280][247, 280][248, 280][249, 280][250, 280][251, 280][252, 280][253, 280][254, 280][255, 280][256, 280][257, 280][258, 280][259, 280][260, 280][261, 280][262, 280][263, 280][264, 280][265, 280][266, 280][267, 280][268, 280][269, 280][270, 280][271, 280][272, 280] [272, 257][272, 256][272, 255][272, 254][272, 253][272, 252][272, 251][272, 250][272, 249][272, 248][272, 247][272, 246][272, 245][272, 244][272, 243][272, 242][272, 241][272, 240][271, 240][270, 240][270, 239][270, 238][270, 237][270, 236][270, 235][270, 234] [263, 340][264, 340][264, 341][264, 342][265, 342][266, 342][266, 343][266, 344][267, 344][268, 344][269, 344][270, 344][270, 345][270, 346][271, 346][272, 346][273, 346][274, 346][275, 346][276, 346][277, 346][278, 346] [310, 638][311, 638][312, 638][313, 638][314, 638][315, 638][316, 638][317, 638][318, 638][319, 638][320, 638][321, 638][322, 638][323, 638][324, 638][325, 638][326, 638][327, 638][328, 638][329, 638][330, 638] [344, 614][344, 613][344, 612][344, 611][344, 610][344, 609][344, 608][344, 607][344, 606][344, 605][344, 604][344, 603][344, 602][344, 601][344, 600][344, 599][344, 598][344, 597][344, 596][344, 595][344, 594][344, 593][344, 592][344, 591][344, 590] [298, 362][299, 362][300, 362][301, 362][302, 362][303, 362][304, 362][305, 362][306, 362][307, 362][308, 362][309, 362][310, 362][310, 361][310, 360][311, 360][312, 360][313, 360][314, 360][315, 360][316, 360] [290, 360][291, 360][292, 360][293, 360][294, 360][295, 360][296, 360][297, 360][298, 360][299, 360][300, 360][301, 360][302, 360][303, 360][304, 360][305, 360][306, 360][307, 360][308, 360][308, 359][308, 358][309, 358][310, 358][311, 358][312, 358][313, 358][314, 358][315, 358][316, 358][317, 358][318, 358] [208, 265][208, 264][209, 264][210, 264][211, 264][212, 264][213, 264][214, 264][215, 264][216, 264][217, 264][218, 264][219, 264][220, 264][221, 264][222, 264][222, 265][222, 266][223, 266][224, 266][225, 266][226, 266] [284, 334][285, 334][286, 334][286, 335][286, 336][287, 336][288, 336][289, 336][290, 336][291, 336][292, 336][292, 337][292, 338][292, 339][292, 340][292, 341][292, 342][293, 342][294, 342][295, 342][296, 342][297, 342][298, 342] [258, 411][258, 410][259, 410][260, 410][261, 410][262, 410][263, 410][264, 410][265, 410][266, 410][267, 410][268, 410][269, 410][270, 410][271, 410][272, 410][273, 410][274, 410][275, 410][276, 410][277, 410][278, 410][279, 410][280, 410][281, 410][282, 410][283, 410][284, 410][285, 410][286, 410] [204, 288][205, 288][206, 288][207, 288][208, 288][209, 288][210, 288][211, 288][212, 288][213, 288][214, 288][215, 288][216, 288][217, 288][218, 288][219, 288][220, 288][221, 288][222, 288][223, 288][224, 288][225, 288][226, 288][227, 288][228, 288][229, 288][230, 288][231, 288][232, 288][233, 288][234, 288][235, 288][236, 288][237, 288][238, 288][239, 288][240, 288] [332, 572][332, 571][332, 570][333, 570][334, 570][335, 570][336, 570][336, 571][336, 572][337, 572][338, 572][339, 572][340, 572][340, 573][340, 574][340, 575][340, 576][341, 576][342, 576][343, 576][344, 576][345, 576][346, 576][347, 576][348, 576] [241, 312][242, 312][243, 312][244, 312][245, 312][246, 312][247, 312][248, 312][249, 312][250, 312][251, 312][252, 312][253, 312][254, 312][255, 312][256, 312][257, 312][258, 312][259, 312][260, 312][261, 312][262, 312] [356, 584][357, 584][358, 584][359, 584][360, 584][361, 584][362, 584][363, 584][364, 584][364, 583][364, 582][364, 581][364, 580][364, 579][364, 578][364, 577][364, 576][364, 575][364, 574][364, 573][364, 572] [265, 592][266, 592][267, 592][268, 592][269, 592][270, 592][271, 592][272, 592][273, 592][274, 592][275, 592][276, 592][277, 592][278, 592][278, 591][278, 590][278, 589][278, 588][278, 587][278, 586][278, 585][278, 584][278, 583][278, 582][278, 581][278, 580][278, 579][278, 578][278, 577][278, 576] [270, 320][271, 320][272, 320][273, 320][274, 320][275, 320][276, 320][277, 320][278, 320][279, 320][280, 320][281, 320][282, 320][283, 320][284, 320][285, 320][286, 320][287, 320][288, 320][289, 320][290, 320] [314, 620][315, 620][316, 620][317, 620][318, 620][319, 620][320, 620][321, 620][322, 620][323, 620][324, 620][325, 620][326, 620][327, 620][328, 620][329, 620][330, 620][331, 620][332, 620][333, 620][334, 620][335, 620][336, 620][337, 620][338, 620] [356, 584][357, 584][358, 584][359, 584][360, 584][361, 584][362, 584][363, 584][364, 584][364, 583][364, 582][364, 581][364, 580][364, 579][364, 578][364, 577][364, 576][364, 575][364, 574][364, 573][364, 572] [236, 177][236, 176][236, 175][236, 174][236, 173][236, 172][236, 171][236, 170][236, 169][236, 168][236, 167][236, 166][235, 166][234, 166][234, 165][234, 164][234, 163][234, 162][234, 161][234, 160][234, 159][234, 158][234, 157][234, 156][234, 155][234, 154][234, 153][234, 152][233, 152][232, 152][232, 151][232, 150] [204, 288][205, 288][206, 288][207, 288][208, 288][209, 288][210, 288][211, 288][212, 288][213, 288][214, 288][215, 288][216, 288][217, 288][218, 288][219, 288][220, 288][221, 288][222, 288][223, 288][224, 288][225, 288][226, 288][227, 288][228, 288][229, 288][230, 288][231, 288][232, 288][233, 288][234, 288][235, 288][236, 288][237, 288][238, 288][239, 288][240, 288] [222, 316][222, 317][222, 318][223, 318][224, 318][225, 318][226, 318][227, 318][228, 318][229, 318][230, 318][231, 318][232, 318][233, 318][234, 318][235, 318][236, 318][237, 318][238, 318][239, 318][240, 318][241, 318][242, 318][243, 318][244, 318][245, 318][246, 318][247, 318][248, 318][249, 318][250, 318][251, 318][252, 318][253, 318][254, 318][255, 318][256, 318][257, 318][258, 318][259, 318][260, 318][261, 318][262, 318][262, 317][262, 316][262, 315][262, 314][262, 313][262, 312] [258, 444][258, 445][258, 446][258, 447][258, 448][258, 449][258, 450][258, 451][258, 452][258, 453][258, 454][259, 454][260, 454][260, 455][260, 456][261, 456][262, 456][263, 456][264, 456][265, 456][266, 456][266, 457][266, 458][267, 458][268, 458][269, 458][270, 458] [224, 258][225, 258][226, 258][227, 258][228, 258][229, 258][230, 258][231, 258][232, 258][233, 258][234, 258][235, 258][236, 258][236, 257][236, 256][237, 256][238, 256][238, 255][238, 254][239, 254][240, 254][241, 254][242, 254][242, 253][242, 252][243, 252][244, 252][245, 252][246, 252][247, 252][248, 252][248, 251][248, 250][248, 249][248, 248][248, 247][248, 246] [228, 253][228, 252][229, 252][230, 252][231, 252][232, 252][232, 251][232, 250][233, 250][234, 250][235, 250][236, 250][237, 250][238, 250][239, 250][240, 250][241, 250][242, 250][243, 250][244, 250][245, 250][246, 250][247, 250][248, 250][248, 249][248, 248][248, 247][248, 246] [195, 296][196, 296][196, 297][196, 298][196, 299][196, 300][197, 300][198, 300][199, 300][200, 300][201, 300][202, 300][203, 300][204, 300][205, 300][206, 300][207, 300][208, 300][209, 300][210, 300][211, 300][212, 300][213, 300][214, 300][215, 300][216, 300][217, 300][218, 300][219, 300][220, 300][221, 300][222, 300][223, 300][224, 300] [222, 258][222, 257][222, 256][222, 255][222, 254][222, 253][222, 252][223, 252][224, 252][225, 252][226, 252][227, 252][228, 252][229, 252][230, 252][231, 252][232, 252][232, 251][232, 250][233, 250][234, 250][235, 250][236, 250][237, 250][238, 250][239, 250][240, 250][241, 250][242, 250][243, 250][244, 250][245, 250][246, 250][247, 250][248, 250][248, 249][248, 248][248, 247][248, 246] [247, 284][248, 284][249, 284][250, 284][251, 284][252, 284][253, 284][254, 284][255, 284][256, 284][257, 284][258, 284][259, 284][260, 284][261, 284][262, 284][263, 284][264, 284][265, 284][266, 284][267, 284][268, 284] [318, 659][318, 658][319, 658][320, 658][321, 658][322, 658][323, 658][324, 658][325, 658][326, 658][327, 658][328, 658][329, 658][330, 658][331, 658][332, 658][333, 658][334, 658][335, 658][336, 658][337, 658][338, 658] [283, 298][284, 298][285, 298][286, 298][287, 298][288, 298][289, 298][290, 298][291, 298][292, 298][293, 298][294, 298][295, 298][296, 298][297, 298][298, 298][299, 298][300, 298][301, 298][302, 298][303, 298][304, 298] [294, 584][295, 584][296, 584][297, 584][298, 584][299, 584][300, 584][301, 584][302, 584][303, 584][304, 584][305, 584][306, 584][306, 583][306, 582][307, 582][308, 582][309, 582][310, 582][311, 582][312, 582][313, 582][314, 582] [258, 578][259, 578][260, 578][261, 578][262, 578][263, 578][264, 578][265, 578][266, 578][267, 578][268, 578][269, 578][270, 578][271, 578][272, 578][273, 578][274, 578][275, 578][276, 578][276, 577][276, 576][277, 576][278, 576] [279, 594][280, 594][280, 595][280, 596][280, 597][280, 598][280, 599][280, 600][280, 601][280, 602][280, 603][280, 604][281, 604][282, 604][283, 604][284, 604][285, 604][286, 604][287, 604][288, 604][289, 604][290, 604][291, 604][292, 604] [334, 684][335, 684][336, 684][337, 684][338, 684][339, 684][340, 684][341, 684][342, 684][343, 684][344, 684][345, 684][346, 684][347, 684][348, 684][349, 684][350, 684][351, 684][352, 684][353, 684][354, 684] [194, 290][195, 290][196, 290][196, 291][196, 292][196, 293][196, 294][196, 295][196, 296][196, 297][196, 298][196, 299][196, 300][197, 300][198, 300][199, 300][200, 300][201, 300][202, 300][203, 300][204, 300][205, 300][206, 300][207, 300][208, 300][209, 300][210, 300][211, 300][212, 300][213, 300][214, 300][215, 300][216, 300][217, 300][218, 300][219, 300][220, 300][221, 300][222, 300][223, 300][224, 300] [244, 296][245, 296][246, 296][247, 296][248, 296][249, 296][250, 296][251, 296][252, 296][253, 296][254, 296][255, 296][256, 296][256, 295][256, 294][256, 293][256, 292][257, 292][258, 292][259, 292][260, 292][261, 292][262, 292][262, 291][262, 290] [288, 646][289, 646][290, 646][291, 646][292, 646][293, 646][294, 646][295, 646][296, 646][297, 646][298, 646][298, 645][298, 644][299, 644][300, 644][300, 643][300, 642][301, 642][302, 642][302, 641][302, 640][303, 640][304, 640][304, 639][304, 638][305, 638][306, 638][306, 637][306, 636] [194, 290][195, 290][196, 290][196, 291][196, 292][196, 293][196, 294][196, 295][196, 296][196, 297][196, 298][196, 299][196, 300][197, 300][198, 300][199, 300][200, 300][201, 300][202, 300][203, 300][204, 300][205, 300][206, 300][207, 300][208, 300][209, 300][210, 300][211, 300][212, 300][213, 300][214, 300][215, 300][216, 300][217, 300][218, 300][219, 300][220, 300][221, 300][222, 300][223, 300][224, 300] [244, 378][244, 379][244, 380][245, 380][246, 380][247, 380][248, 380][249, 380][250, 380][251, 380][252, 380][253, 380][254, 380][255, 380][256, 380][257, 380][258, 380][258, 381][258, 382][258, 383][258, 384][259, 384][260, 384] [230, 255][230, 254][230, 253][230, 252][231, 252][232, 252][232, 251][232, 250][233, 250][234, 250][235, 250][236, 250][237, 250][238, 250][239, 250][240, 250][241, 250][242, 250][243, 250][244, 250][245, 250][246, 250][247, 250][248, 250][248, 249][248, 248][248,

247][248, 246] [246, 229][246, 228][247, 228][248, 228][249, 228][250, 228][251,
228][252, 228][253, 228][254, 228][255, 228][256, 228][257, 228][258, 228][259,
228][260, 228][261, 228][262, 228][263, 228][264, 228][265, 228][266, 228][267,
228][268, 228][269, 228][270, 228][271, 228][272, 228][273, 228][274, 228][275,
228][276, 228][277, 228][278, 228][279, 228][280, 228] [244, 264][245, 264][246,
264][247, 264][248, 264][249, 264][250, 264][250, 263][250, 262][251, 262][252,
262][253, 262][254, 262][255, 262][256, 262][257, 262][258, 262][259, 262][260,
262][261, 262][262, 262][263, 262][264, 262][264, 261][264, 260][265, 260][266,
260] [229, 262][230, 262][231, 262][232, 262][233, 262][234, 262][235, 262][236,
262][237, 262][238, 262][239, 262][240, 262][241, 262][242, 262][243, 262][244,
262][245, 262][246, 262][247, 262][248, 262][249, 262][250, 262][251, 262][252,
262][253, 262][254, 262][255, 262][256, 262][257, 262][258, 262][259, 262][260,
262][261, 262][262, 262][263, 262][264, 262][264, 261][264, 260][265, 260][266,
260] [195, 296][196, 296][196, 297][196, 298][196, 299][196, 300][197, 300][198,
300][199, 300][200, 300][201, 300][202, 300][203, 300][204, 300][205, 300][206,
300][207, 300][208, 300][209, 300][210, 300][211, 300][212, 300][213, 300][214,
300][215, 300][216, 300][217, 300][218, 300][219, 300][220, 300][221, 300][222,
300][223, 300][224, 300] [242, 302][243, 302][244, 302][245, 302][246, 302][247,
302][248, 302][249, 302][250, 302][251, 302][252, 302][253, 302][254, 302][255,
302][256, 302][257, 302][258, 302][258, 303][258, 304][259, 304][260, 304] [262,
505][262, 504][263, 504][264, 504][265, 504][266, 504][267, 504][268, 504][269,
504][270, 504][271, 504][272, 504][273, 504][274, 504][275, 504][276, 504][276,
505][276, 506][277, 506][278, 506][279, 506][280, 506][281, 506][282, 506][283,
506][284, 506][285, 506][286, 506][287, 506][288, 506][289, 506][290, 506][291,
506][292, 506] [294, 613][294, 612][295, 612][296, 612][297, 612][298, 612][299,
612][300, 612][301, 612][302, 612][302, 613][302, 614][303, 614][304, 614][305,
614][306, 614][307, 614][308, 614][309, 614][310, 614][311, 614][312, 614] [328,
688][329, 688][330, 688][331, 688][332, 688][333, 688][334, 688][335, 688][336,
688][337, 688][338, 688][339, 688][340, 688][341, 688][342, 688][343, 688][344,
688][345, 688][346, 688][347, 688][348, 688][348, 689][348, 690][349, 690][350,
690][351, 690][352, 690] [290, 615][290, 614][291, 614][292, 614][293, 614][294,
614][295, 614][296, 614][297, 614][298, 614][299, 614][300, 614][301, 614][302,
614][303, 614][304, 614][305, 614][306, 614][307, 614][308, 614][309, 614][310,
614][311, 614][312, 614] [266, 486][267, 486][268, 486][269, 486][270, 486][271,
486][272, 486][273, 486][274, 486][275, 486][276, 486][277, 486][278, 486][279,
486][280, 486][281, 486][282, 486][283, 486][284, 486][285, 486][286, 486] [284,
617][284, 616][285, 616][286, 616][287, 616][288, 616][289, 616][290, 616][291,
616][292, 616][293, 616][294, 616][295, 616][296, 616][297, 616][298, 616][299,
616][300, 616][301, 616][302, 616][303, 616][304, 616][305, 616][306, 616][307,
616][308, 616][309, 616][310, 616] [224, 255][224, 254][225, 254][226, 254][227,
254][228, 254][229, 254][230, 254][230, 253][230, 252][231, 252][232, 252][232,
251][232, 250][233, 250][234, 250][235, 250][236, 250][237, 250][238, 250][239,
250][240, 250][241, 250][242, 250][243, 250][244, 250][245, 250][246, 250][247,
250][248, 250][248, 249][248, 248][248, 247][248, 246] [310, 442][311, 442][312,
442][313, 442][314, 442][315, 442][316, 442][317, 442][318, 442][319, 442][320,
442][321, 442][322, 442][323, 442][324, 442][325, 442][326, 442][327, 442][328,

442][329, 442][330, 442][330, 441][330, 440][330, 439][330, 438][330, 437][330, 436][330, 435][330, 434][330, 433][330, 432] [281, 282][282, 282][283, 282][284, 282][285, 282][286, 282][287, 282][288, 282][289, 282][290, 282][291, 282][292, 282][293, 282][294, 282][295, 282][296, 282][297, 282][298, 282][299, 282][300, 282][301, 282][302, 282] [254, 249][254, 248][255, 248][256, 248][257, 248][258, 248][259, 248][260, 248][260, 247][260, 246][260, 245][260, 244][260, 243][260, 242][261, 242][262, 242][263, 242][264, 242][265, 242][266, 242][266, 241][266, 240] [254, 249][254, 248][255, 248][256, 248][257, 248][258, 248][259, 248][260, 248][260, 247][260, 246][260, 245][260, 244][260, 243][260, 242][261, 242][262, 242][263, 242][264, 242][265, 242][266, 242][266, 241][266, 240] [294, 613][294, 612][295, 612][296, 612][297, 612][298, 612][299, 612][300, 612][301, 612][302, 612][302, 613][302, 614][303, 614][304, 614][305, 614][306, 614][307, 614][308, 614][309, 614][310, 614][311, 614][312, 614] [254, 249][254, 248][255, 248][256, 248][257, 248][258, 248][259, 248][260, 248][260, 247][260, 246][260, 245][260, 244][260, 243][260, 242][261, 242][262, 242][263, 242][264, 242][265, 242][266, 242][266, 241][266, 240] [258, 433][258, 432][258, 431][258, 430][258, 429][258, 428][259, 428][260, 428][261, 428][262, 428][263, 428][264, 428][265, 428][266, 428][267, 428][268, 428][269, 428][270, 428][271, 428][272, 428][273, 428][274, 428] [254, 249][254, 248][255, 248][256, 248][257, 248][258, 248][259, 248][260, 248][260, 247][260, 246][260, 245][260, 244][260, 243][260, 242][261, 242][262, 242][263, 242][264, 242][265, 242][266, 242][266, 241][266, 240] [254, 249][254, 248][255, 248][256, 248][257, 248][258, 248][259, 248][260, 248][260, 247][260, 246][260, 245][260, 244][260, 243][260, 242][261, 242][262, 242][263, 242][264, 242][265, 242][266, 242][266, 241][266, 240] [290, 615][290, 614][291, 614][292, 614][293, 614][294, 614][295, 614][296, 614][297, 614][298, 614][299, 614][300, 614][301, 614][302, 614][303, 614][304, 614][305, 614][306, 614][307, 614][308, 614][309, 614][310, 614][311, 614][312, 614] [254, 249][254, 248][255, 248][256, 248][257, 248][258, 248][259, 248][260, 248][260, 247][260, 246][260, 245][260, 244][260, 243][260, 242][261, 242][262, 242][263, 242][264, 242][265, 242][266, 242][266, 241][266, 240] [242, 224][242, 225][242, 226][242, 227][242, 228][243, 228][244, 228][245, 228][246, 228][247, 228][248, 228][249, 228][250, 228][251, 228][252, 228][253, 228][254, 228][255, 228][256, 228][257, 228][258, 228][259, 228][260, 228][261, 228][262, 228][263, 228][264, 228][265, 228][266, 228][267, 228][268, 228][269, 228][270, 228][271, 228][272, 228][273, 228][274, 228][275, 228][276, 228][277, 228][278, 228][279, 228][280, 228] [346, 432][346, 431][346, 430][346, 429][346, 428][346, 427][346, 426][346, 425][346, 424][346, 423][346, 422][346, 421][346, 420][346, 419][346, 418][346, 417][346, 416][346, 415][346, 414][346, 413][346, 412][346, 411][346, 410][345, 410][344, 410] [346, 432][346, 431][346, 430][346, 429][346, 428][346, 427][346, 426][346, 425][346, 424][346, 423][346, 422][346, 421][346, 420][346, 419][346, 418][346, 417][346, 416][346, 415][346, 414][346, 413][346, 412][346, 411][346, 410][345, 410][344, 410] [294, 256][294, 257][294, 258][295, 258][296, 258][296, 259][296, 260][297, 260][298, 260][298, 261][298, 262][299, 262][300, 262][301, 262][302, 262][303, 262][304, 262][305, 262][306, 262][307, 262][308, 262][309, 262][310, 262][310, 261][310, 260][311, 260][312, 260][312, 259][312, 258][313, 258][314, 258] [310, 320][311, 320][312, 320][312, 321][312, 322][313, 322][314, 322][314, 323][314, 324][315,

324][316, 324][316, 325][316, 326][316, 327][316, 328][316, 329][316, 330][317,
330][318, 330][318, 331][318, 332][319, 332][320, 332][321, 332][322, 332] [302,
314][301, 314][300, 314][299, 314][298, 314][298, 313][298, 312][298, 311][298,
310][299, 310][300, 310][300, 309][300, 308][301, 308][302, 308][303, 308][304,
308][304, 307][304, 306][305, 306][306, 306][307, 306][308, 306][309, 306][310,
306] [304, 246][304, 247][304, 248][304, 249][304, 250][305, 250][306, 250][307,
250][308, 250][308, 249][308, 248][309, 248][310, 248][311, 248][312, 248][313,
248][314, 248][315, 248][316, 248][316, 249][316, 250][317, 250][318, 250] [242,
336][243, 336][244, 336][245, 336][246, 336][247, 336][248, 336][249, 336][250,
336][251, 336][252, 336][253, 336][254, 336][255, 336][256, 336][257, 336][258,
336][259, 336][260, 336][261, 336][262, 336] [244, 328][244, 327][244, 326][244,
325][244, 324][244, 323][244, 322][245, 322][246, 322][247, 322][248, 322][249,
322][250, 322][251, 322][252, 322][253, 322][254, 322][255, 322][256, 322][257,
322][258, 322][259, 322][260, 322][261, 322][262, 322][262, 321][262, 320] [298,
258][299, 258][300, 258][301, 258][302, 258][302, 259][302, 260][303, 260][304,
260][305, 260][306, 260][307, 260][308, 260][309, 260][310, 260][311, 260][312,
260][312, 259][312, 258][313, 258][314, 258] [272, 422][273, 422][274, 422][274,
421][274, 420][275, 420][276, 420][276, 419][276, 418][275, 418][274, 418][273,
418][272, 418][272, 417][272, 416][272, 415][272, 414][273, 414][274, 414][274,
413][274, 412] [242, 320][243, 320][244, 320][245, 320][246, 320][247, 320][248,
320][249, 320][250, 320][251, 320][252, 320][253, 320][254, 320][255, 320][256,
320][257, 320][258, 320][259, 320][260, 320][261, 320][262, 320] [242, 324][242,
323][242, 322][243, 322][244, 322][245, 322][246, 322][247, 322][248, 322][249,
322][250, 322][251, 322][252, 322][253, 322][254, 322][255, 322][256, 322][257,
322][258, 322][259, 322][260, 322][261, 322][262, 322][262, 321][262, 320] [306,
470][307, 470][308, 470][308, 471][308, 472][308, 473][308, 474][309, 474][310,
474][311, 474][312, 474][313, 474][314, 474][315, 474][316, 474][317, 474][318,
474][319, 474][320, 474][321, 474][322, 474] [301, 368][302, 368][302, 369][302,
370][302, 371][302, 372][303, 372][304, 372][304, 373][304, 374][305, 374][306,
374][306, 375][306, 376][306, 377][306, 378][307, 378][308, 378][308, 379][308,
380][309, 380][310, 380] [310, 266][311, 266][312, 266][312, 267][312, 268][313,
268][314, 268][314, 269][314, 270][315, 270][316, 270][317, 270][318, 270][319,
270][320, 270][321, 270][322, 270][323, 270][324, 270][325, 270][326, 270][327,
270] [300, 260][300, 261][300, 262][301, 262][302, 262][303, 262][304, 262][305,
262][306, 262][307, 262][308, 262][309, 262][310, 262][310, 261][310, 260][311,
260][312, 260][312, 259][312, 258][313, 258][314, 258] [294, 281][294, 280][295,
280][296, 280][297, 280][298, 280][298, 279][298, 278][299, 278][300, 278][300,
277][300, 276][300, 275][300, 274][300, 273][300, 272][301, 272][302, 272][303,
272][304, 272][305, 272][306, 272][307, 272][308, 272][309, 272][310, 272] [308,
280][309, 280][310, 280][311, 280][312, 280][313, 280][314, 280][314, 279][314,
278][314, 277][314, 276][315, 276][316, 276][317, 276][318, 276][319, 276][320,
276][321, 276][322, 276][323, 276][324, 276] [266, 230][267, 230][268, 230][269,
230][270, 230][271, 230][272, 230][273, 230][274, 230][275, 230][276, 230][277,
230][278, 230][279, 230][280, 230][281, 230][282, 230][282, 229][282, 228][283,
228][284, 228][284, 227][284, 226][285, 226][286, 226][286, 225][286, 224][286,
223][286, 222][287, 222][288, 222][288, 221] [260, 384][261, 384][262, 384][263,

384][264, 384][264, 385][264, 386][265, 386][266, 386][267, 386][268, 386][269, 386][270, 386][271, 386][272, 386][273, 386][274, 386][274, 385][274, 384][275, 384][276, 384] [266, 218][266, 219][266, 220][267, 220][268, 220][269, 220][270, 220][271, 220][272, 220][273, 220][274, 220][275, 220][276, 220][277, 220][278, 220][279, 220][280, 220][281, 220][282, 220][283, 220][284, 220][285, 220][286, 220][287, 220] [242, 324][242, 323][242, 322][243, 322][244, 322][245, 322][246, 322][247, 322][248, 322][249, 322][250, 322][251, 322][252, 322][253, 322][254, 322][255, 322][256, 322][257, 322][258, 322][259, 322][260, 322][261, 322][262, 322][262, 321][262, 320] [266, 230][267, 230][268, 230][269, 230][270, 230][271, 230][272, 230][273, 230][274, 230][275, 230][276, 230][277, 230][278, 230][279, 230][280, 230][281, 230][282, 230][282, 229][282, 228][283, 228][284, 228][284, 227][284, 226][285, 226][286, 226][286, 225][286, 224][286, 223][286, 222][287, 222][288, 222][288, 221] [268, 226][269, 226][270, 226][271, 226][272, 226][273, 226][274, 226][275, 226][276, 226][277, 226][278, 226][279, 226][280, 226][281, 226][282, 226][283, 226][284, 226][285, 226][286, 226][286, 225][286, 224][286, 223][286, 222][287, 222][288, 222][288, 221] [268, 232][269, 232][270, 232][271, 232][272, 232][273, 232][274, 232][275, 232][276, 232][277, 232][278, 232][278, 231][278, 230][279, 230][280, 230][281, 230][282, 230][282, 229][282, 228][283, 228][284, 228][284, 227][284, 226][285, 226][286, 226][286, 225][286, 224][286, 223][286, 222][287, 222][288, 222][288, 221] [306, 227][306, 226][306, 225][306, 224][307, 224][308, 224][308, 223][308, 222][309, 222][310, 222][310, 221][310, 220][310, 219][310, 218][310, 217][310, 216][310, 215][310, 214][310, 213][310, 212][310, 211] [304, 227][304, 226][305, 226][306, 226][306, 225][306, 224][307, 224][308, 224][308, 223][308, 222][309, 222][310, 222][310, 221][310, 220][310, 219][310, 218][310, 217][310, 216][310, 215][310, 214][310, 213][310, 212][310, 211] [304, 408][305, 408][306, 408][307, 408][308, 408][309, 408][310, 408][311, 408][312, 408][313, 408][314, 408][315, 408][316, 408][317, 408][318, 408][319, 408][320, 408][321, 408][322, 408][323, 408][324, 408][325, 408][326, 408][327, 408][328, 408][328, 407][328, 406] [296, 279][296, 278][297, 278][298, 278][299, 278][300, 278][300, 277][300, 276][300, 275][300, 274][300, 273][300, 272][301, 272][302, 272][303, 272][304, 272][305, 272][306, 272][307, 272][308, 272][309, 272][310, 272] [266, 272][266, 271][266, 270][266, 269][266, 268][266, 267][266, 266][267, 266][268, 266][269, 266][270, 266][270, 267][270, 268][271, 268][272, 268][273, 268][274, 268][275, 268][276, 268][276, 269][276, 270] [349, 614][350, 614][350, 615][350, 616][350, 617][350, 618][350, 619][350, 620][350, 621][350, 622][351, 622][352, 622][352, 623][352, 624][352, 625][352, 626][351, 626][350, 626][350, 627][350, 628][350, 629][350, 630][350, 631][350, 632][350, 633][350, 634] [292, 282][292, 281][292, 280][293, 280][294, 280][295, 280][296, 280][297, 280][298, 280][298, 279][298, 278][299, 278][300, 278][300, 277][300, 276][300, 275][300, 274][300, 273][300, 272][301, 272][302, 272][303, 272][304, 272][305, 272][306, 272][307, 272][308, 272][309, 272][310, 272] [307, 406][308, 406][308, 407][308, 408][309, 408][310, 408][311, 408][312, 408][313, 408][314, 408][315, 408][316, 408][317, 408][318, 408][319, 408][320, 408][321, 408][322, 408][323, 408][324, 408][325, 408][326, 408][327, 408][328, 408][328, 407][328, 406] [340, 440][341, 440][342, 440][343, 440][344, 440][345, 440][346, 440][347, 440][348, 440][349, 440][350, 440][350, 441][350, 442][350, 443][350, 444][351, 444][352, 444][353, 444][354, 444][355, 444][356, 444][357, 444][358, 444] [315, 338][316, 338][317, 338][318, 338][319, 338][320, 338][321, 338][322, 338][323, 338][324, 338][325, 338][326, 338][327, 338][328, 338][329, 338][330, 338][331, 338][332, 338][333, 338][334, 338][334, 337][334, 336] [304, 277][304, 276][305, 276][306, 276][307, 276][308, 276][309, 276][310, 276][311, 276][312, 276][313, 276][314, 276][315, 276][316, 276][317, 276][318, 276][319, 276][320, 276][321, 276][322, 276][323, 276][324, 276] [300, 340][301, 340][302, 340][302, 339][302, 338][303, 338][304, 338][305, 338][306, 338][307, 338][308, 338][309, 338][310, 338][311, 338][312, 338][313, 338][314, 338][314, 339][314, 340][315, 340][316, 340][317, 340][318, 340][319, 340][320, 340][321, 340][322, 340][323, 340][324, 340][325, 340][326, 340] [340, 440][341, 440][342, 440][343, 440][344, 440][345, 440][346, 440][347, 440][348, 440][349, 440][350, 440][350, 441][350, 442][350, 443][350, 444][351, 444][352, 444][353, 444][354, 444][355, 444][356, 444][357, 444][358, 444] [306, 227][306, 226][306, 225][306, 224][307, 224][308, 224][308, 223][308, 222][309, 222][310, 222][310, 221][310, 220][310, 219][310, 218][310, 217][310, 216][310, 215][310, 214][310, 213][310, 212][310, 211] [259, 382][260, 382][261, 382][262, 382][262, 383][262, 384][263, 384][264, 384][264, 385][264, 386][265, 386][266, 386][267, 386][268, 386][269, 386][270, 386][271, 386][272, 386][273, 386][274, 386][274, 385][274, 384][275, 384][276, 384] [315, 338][316, 338][317, 338][318, 338][319, 338][320, 338][321, 338][322, 338][323, 338][324, 338][325, 338][326, 338][327, 338][328, 338][329, 338][330, 338][331, 338][332, 338][333, 338][334, 338][334, 337][334, 336] [315, 338][316, 338][317, 338][318, 338][319, 338][320, 338][321, 338][322, 338][323, 338][324, 338][325, 338][326, 338][327, 338][328, 338][329, 338][330, 338][331, 338][332, 338][333, 338][334, 338][334, 337][334, 336] [300, 334][301, 334][302, 334][302, 335][302, 336][302, 337][302, 338][303, 338][304, 338][305, 338][306, 338][307, 338][308, 338][309, 338][310, 338][311, 338][312, 338][313, 338][314, 338][314, 339][314, 340][315, 340][316, 340][317, 340][318, 340][319, 340][320, 340][321, 340][322, 340][323, 340][324, 340][325, 340][326, 340] [302, 276][302, 275][302, 274][303, 274][304, 274][305, 274][306, 274][307, 274][308, 274][309, 274][310, 274][311, 274][312, 274][313, 274][314, 274][315, 274][316, 274][317, 274][318, 274][319, 274][320, 274][321, 274][322, 274][323, 274][324, 274] [294, 294][294, 293][294, 292][294, 291][294, 290][294, 289][294, 288][295, 288][296, 288][297, 288][298, 288][299, 288][300, 288][301, 288][302, 288][303, 288][304, 288][305, 288][306, 288][306, 287][306, 286] [280, 480][279, 480][278, 480][278, 479][278, 478][278, 477][278, 476][279, 476][280, 476][281, 476][282, 476][283, 476][284, 476][285, 476][286, 476][287, 476][288, 476][289, 476][290, 476][290, 475][290, 474] [242, 338][243, 338][244, 338][245, 338][246, 338][247, 338][248, 338][249, 338][250, 338][251, 338][252, 338][253, 338][254, 338][255, 338][256, 338][257, 338][258, 338][259, 338][260, 338][261, 338][262, 338]