אלגוריתם למציאת המסלול הקצר ביותר:

רקע:

בהינתן 2 נקודות על מפה עם קרחונים, נרצה למצוא את המסלול הקצר ביותר בין שתי הנקודות שלא עובר דרך הקרחונים. תמונה לדוגמה:

1. חפשי וכתבי בקצרה על 3 אלגוריתמי חיפוש עיקריים (אופטימליים)

דייקסטרה וAstar

1. יצירת המפה בפייתון: כתבי סקריפט בפייתון המגריל גודל מפה, מספר פוליגונים, רדיוס נקודות ומספר נקודות לכל פוליגון (את הפוליגון תחילה אנחנו יוצרים על ידי הגרלת מספר נקודות ברדיוס מסוים). כתבי את כל הנתונים לקובץ טקסט בסדר הבא (רק מספרים, העברית היא בשביל ההסבר):

300 –(גודל מפה)

5 –(מספר הקרחונים)

1 – (קרחון מס' 1)

3 - (מספר הנקודות בקרחון)

200 110 – (נ.צ נקודה)

201 115 – (נ.צ נקודה)

212 120 – (נ.צ נקודה)

2 – (קרחון מס' 2)

5 - (מספר הנקודות בקרחון)

200 110 – (נ.צ נקודה)

201 115 – (נ.צ נקודה)

212 120 – (נ.צ נקודה)

212 120 – (נ.צ נקודה)

212 120 – (נ.צ נקודה)

וכך הלאה לכל קרחון

1. עבורי ל-c++ וכתבי קוד הקולט את הנתונים לתוך classes ומבני נתונים
2. השתמשי ב-convex hull כדי להפוך את הגושי נקודות לפוליגון
3. בחרי דרך מפה: grid או יצירת גרף (הנקודות הם רק נקודות ההתחלה והסוף ונקודות הפוליגונים).

**תוכנית:**

עבור כל שתי נקודות במפה נבדוק האם יש line of sight.

איך? נבדוק האם יש פוליגון ביניהם.

איך? עבור כל פוליגון נבדוק האם כל שתי נקודות נחתכות עם 2 הנקודות מהמפה, אם כן אז נתקענו בפוליגון. נשמור את המסלולים "הטובים" בין כל שתי נקודות ונקבל מסלול בין שתי נקודות. כלומר עבור כל נקודה נשמור מערך של שכנים.

**פסודו:**

1. לולאה על כל הפוליגונים ואז על כל הנקודות - היא בשביל לעבור על כל הנקודות ( לכאורה לבחור את ה00)
2. לולאה על כל הפוליגונים ואז על כל הנקודות- מציאת שכן.
3. למצוא האם נחתך ע"י נקודה אחרת:
   1. לולאה על כל הפוליגונים ואז על כל הנקודות
   2. לולאה על כל הפוליגונים ואז על כל הנקודות
4. הפעלי אלגוריתם חיפוש אחד שבחרת מבין השלושה שפירטת עליהם בסעיף 1
5. החזרי את הנתונים לפייתון והציגי ויזואלית את המסלול והפוליגונים!

התוצר אמור להיראות כך:

1. סעיף בונוס א': תייצרי GUI בפייתון אינטרקטיבי
2. סעיף בונוס ב': השתמש ב-cmakelist / כתוב סקריפט / כל דרך אחרת שבלחיצת כפתור כל השלבים מופעלים ביחד.