המספר הבינארי מייצג את מי עברנו במסלול

לדוגמא 7 = 0111

זה אומר שעברנו בקודקודים 1,2,3

נתחיל מאיתחול המסלולים שעוברים מ0 לקודקוד i כלשהו (כלומר מסלול זול ביותר שמתחיל בקודקוד 0 ועובר בקודקוד i בלבד) למרחק בין אותו קודקוד ל0. זה נתון בטבלה dists שהבאנו מfloyd-warshal.

נבדוק לכל גודל סט אפשרי (מ2 עד n)

ולכל גודל כזה כל קומבינציה אפשרית של nodes שמשתתפים – מה המסלול הזול ביותר העובר בין כל אותם nodes שמשתתפים?

נעשה זאת כך –

המשתנה bits נוצר ע"י פעולת shift ופעולת OR.  
על כל קודקוד, המיוצג ע"י מספר (קודקוד 1, קודקוד 2 וכו') נעשה SHIFT על המספר 1 לפי המספר המייצג את הקודקוד.  
לדוגמא עבור קודקוד 3 נקבל 0001 -> 1000.  
לכל תוצאה כזו נעשה OR עם מה שהיה עד עכשיו.  
כך נקבל ייצוג בינארי בו כל 1 מייצג קודקוד שמשתתף (מי שלא משתתף יהיה 0).

לדוגמא עבור הsubset 1,2,3 נקבל bits = 1110

כעת נרצה לבדוק מה הדרך הזולה ביותר לעבור בsubset הזה.  
נבחר כל פעם קודקוד אחד מה-subset ונבדוק כמה יעלה לעבור בכל שאר הקודקודים ואז לבוא אליו.

נעשה זאת כך –

הקודקוד שנבחר לסיים בו בכל איטרציה של בדיקה ייקרא k.

הערך prev ייצג לנו בייצוג בינארי מי הקודקודים עליהם נעבור קודם (כלומר כל אלו שב-subset חוץ מהקודקוד שנבחר לסיים בו - k.)  
הוא יווצר ע"י זה שנייצר את הייצוג הבינארי של k (ע"י SHIFT של 1 בk), נעשה לתוצאה משלים ל1 (היפוך כל ה1 וה0 בייצוג הבינארי). את התוצאה של כל זה ניקח ונעשה איתה AND עם bits.  
כך למעשה מה שעשינו היה לחסר רק את ה1 של הk שבחרנו מתוך כל הsubset בו אנחנו עוסקים.

לדוגמא –

Subset = 1,2,3,4

Bits = 11110

When k = 4

Prev = bits & ~(1 << k) = (11110) & ~(1 << 4) = (11110) & ~(10000) = (11110) & (01111) = 01110

כלומר יש לעבור בקודקודים 1,2,3, לפני שעוברים ב4, כנדרש.  
(מן הסתם שקודם יוצאים מ0, ובהמשך גם נחזור אליו בסוף המסלול).

אז אנחנו יודעים במי נידרש לעבור קודם prev – ונבקש את המסלול הזול ביותר לכך (שכבר שמרנו בשלב קודם)

אנחנו גם יודעים במי נרצה לסיים (k)

נבנה מערך options בו נשמור את האופציות השונות, מי יהיה הקודקוד האחרון לפני k. נקרא לו m.  
אז m יהיה בכל מעבר על הלולאת for אחד מהקודקודים subset\k (הsubset בלי k)

בכל מעבר על לולאת הfor למילוי options, נבדוק מה העלות הזולה ביותר לעבור בכולם בsubset חוץ מk, כשנסיים בm, נוסיף את העלות מm לk, ונתעד שהקודקוד האחרון במסלול הוא m.  
(את העלות של כל הקודמים כבר חישבנו בשלב קודם, שהרי אנחנו מחשבים כל תת-קבוצה של הקודקודים בגרף, מתחילים בגודל קבוצה קטן ועולים עד לכל הקודקודים)

לאחר סיום הלולאה, נשמור בC את האופציה המינימאלית שחישבנו (כלומר מה הדרך הזולה ביותר לעבור בכולם בsubset ולסיים בk)

נעשה את הנ"ל שוב ושוב לכל גודל קבוצה ולכל אפשרות לאיזה קודקודים יהיו בקבוצה (כלומר כל הדרכים לבחור 2 קודקודים, כל הדרכים לבחור 3 קודקודים וכו')

כך נקבל את כל הדרכים לעבור בכל תת-קבוצה של קודקודי הגרף, במסלול הזול ביותר.

לאחר מכן נבחר באיזה קודקודים אנחנו רוצים לעבור (לפי הקלט)  
נבדוק באיזה קודקוד הכי טוב לסיים את המסלול, כלומר לכל k מבין הקודקודים בהם אנחנו חייבים לעבור – עבור איזה k נקבל את המסלול הזול ביותר שעובר בכל הקודקודים בהם נדרשנו לעבור, מסיים בk וחוזר ל0?

נכניס עבור כל k את התשובה למערך options\_final (לא אותו אחד ממקודם, אחד חדש), ונבחר מביניהם את המינימאלי, ונבקש גם את הקודקוד האחרון.  
ממנו נוכל לשחזר אחורה את המסלול הזול ביותר, ולסיים.

לגבי סעיף ב' –

אין סיבה להוציא 2 מוניות אף פעם כל עוד שאין התחשבות בגורם הזמן, שלא התייחסו אליו בשאלה.

הרי 2 המוניות חייבות לחזור בסוף להתחלה, נגיד שזה 0 בה"כ.

כל מונית עושה מסלול כלשהו

0 [….] 0

במקום להוציא 2 מוניות, אפשר לעשות את המסלול של שתיהן, אחת אחרי השנייה, במונית אחת, באותה העלות.

(כבר כתבתי קוד כדי לבדוק מה האופציה הטובה ביותר לחילוק האנשים ל2 מוניות, רק אחרי הבנתי שלמעשה אין צורך בגלל הנ"ל)