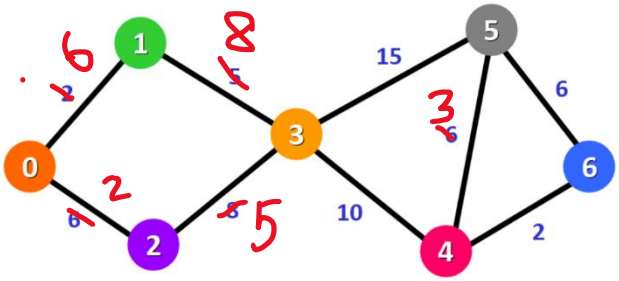
05506005 ชื่อ นายธนกร บุญสินธุ์ รหัสนักศึกษา 67050200

ใบงานที่ 5

วัตถุประสงค์ เพื่อสร้างประสบการณ์การใช้ graph ด้วย adjacency matrix



initial

Distance:
0: 0
1: ∞
2: ∞
3: ∞
4: ∞
5: ∞
6: ∞

city = 0

Distance:
0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ∞
4: ∞
5: ∞
6: ∞

โจทย์ single source shortest path หมายถึง การหาค่าว่าจาก ต้นทาง ไปแต่ละเมืองในแผนที่ (graph) ระยะทางที่สั้นที่สุดเป็นเท่าไหร่

หนึ่งในวิธีการ represent graph คือสร้าง ตาราง 2 มิติ (เรียก adjacenty matrix เพราะ adjacency คืออยู่ติดกัน) โดยระบุระยะทางตาม graph ใส่ค่า infinity สำหรับเมืองที่ไปไม่ได้ (ส่วนของโปรแกรมตัวอย่างใส่ค่า 711 ก็เพียงพอต่อการทดสอบ (เพราะมันแพงกว่าค่าในโจทย์) เช่น adja[1][3] = adja[3][1] = 5 ดูภาพกราฟโจทย์

หลักการคิดคือเราจะพิจารณาตามจำนวนเมือง เช่น รอบแรกพิจารณาจากเมือง 0 (source) โดยเราจะดูทุกเมือง (dest) ว่ามีทางไปได้ไหม adja[0][dest] > 0 (และไม่ใช่ infinity) ก็คือมี edge จาก source ดังนั้นรอบแรกจะมีเพียง 2 เมือง ที่ได้บันทึกในตาราง (เพราะ dest อื่นเป็น infinity)

รอบถัดๆไป คือ เรานำมาพิจารณาว่า dist[city] + adja[city][dest] มีระยะทางสั้นกว่า (จากเท่าที่รอบก่อนๆรู้มา) กล่าวคือหากการเดินทางผ่าน city นี้ ทำให้ทาง dist[dest] ลดลง เราจึง update dist[dest] นึกภาพว่า เรามี dist[สมุทรสงคราม] dist[เพชรบุรี] เมื่อ city เป็น นครปฐม เราก็นำมาคำนวณว่า dist[นครปฐม] + adja[นครปฐม][สมุทรสงคราม] สั้นว่า dist[สมุทรสงคราม] ที่เคยคำนวณไว้หรือไม่

ประเด็นการพิสูจน์ ว่าสุดท้ายแล้วจะได้สั้นสุดทุกเมืองได้จริงหรือเปล่า นั่นเป็นเพราะ ลูปนอก เรานำทุกเมืองมาพิจารณา และลูปใน เรานำเมืองนั้นๆ(dest) มาเทียบกับทุกปลายทาง (ลูปนอกที่หา shortest distance ไปแล้ว)

แอปพลิเคชันแนวนี้คือ หาเส้นทางที่เหมาะที่สุดที่ส่งข้อมูลข้ามอินเตอร์เน็ต

Distance:
0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7 from (5 + 2) vs. 14 from (6 + 8)
4: ∞
5: ∞
6: ∞

city = 1, 2

Distance:
0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7
4: ~~∞~~ 17 from (2 + 5 + 10)
5: ~~∞~~ 22 from (2 + 5 + 15)
6: ∞

city = 3

Distance:
0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7
4: ~~∞~~ 17
5: ~~∞~~ 22 vs. 23 (2 + 5 + 10 + 6)
6: ~~∞~~ 19 from (2 + 5 + 10 + 2)

city = 4

Distance:
0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7
4: ~~∞~~ 17
5: ~~∞~~ 22
6: ~~∞~~ 19

city = 5, 6

ภาพประกอบจาก https://www.freecodecamp.org/ news/dijkstras-shortest-path-algorithm-visual-introduction/ พึงตระหนักว่า dijkstra ของแท้สามารถ implement ได้ด้วย O (E logV) ลำดับการประมวลผลใน page ไม่ตรงกับส่วนของโปรแกรมด้านล่าง เพราะทำการ simplify code

05506005 ชื่อ ..นายธนกร บุญสินธุ์..... รหัสนักศึกษา 67050200.....

```
[][] adja ={{   0,   6,   2, 711, 711, 711, 711 },
            {   6,   0, 711,   8, 711, 711, 711 },
            {   2, 711,   0,   5, 711, 711, 711 },
            { 711,   8,   5,   0,  10,  15, 711 },
            { 711, 711, 711,  10,   0,   3,   2 },
            { 711, 711, 711,  15,   3,   0,   6 },
            { 711, 711, 711, 711,   2,   6,   0 } };

[] dist ={ 0, 711, 711, 711, 711, 711, 711 };

for ( city = 0; city < adja.length; city++) {
  for ( dest = city + 1; dest < adja.length; dest++) {
    if ( /* q1 */ ) {
       tmp++;
  // print("new cost for " + dest + " from " + dist[dest]);
  // println(" to " + (dist[city] + adja[city][dest]) + " via " + city );
       dist[dest] = dist[city] + adja[city][dest];
    }
  } // for dest
  System.out.println("from 0 " + dist[dest]);
  System.out.println("------------------------");
}
System.out.println(tmp); /* q2 */
System.out.println(Arrays.toString(dist)); /* q3 */
```

คำสั่ง

q1 เติมส่วนของโปรแกรม

dist[dest] < 711

q2 แสดงผลการทำงาน

System.out.println(Arrays.toString(dist));

q3 แสดงผลการทำงาน

System.out.println(tmp);

System.out.println(Arrays.toString(dist));

กำหนดส่ง TBA