امیرحسین نجفی زاده توجه: [در حل این سوال فرض شده که از root یا s به همه رئوس حداقل یک مسیر وجود دارد]

۹۸۳۱۰۶۵

$s = Root$

سوال ۵ : الف) I) فرض کنید رأس ورودی S داده شده باشد . تنها فرقی که در این جا داریم این است که تعداد یال ها کمتر هستند . در الگوریتم dijkstra ما یک queue از رأس داریم براساس key ، و روی queue متد Extract-Min را صدا می زنیم . تا به اینجای کار جهت دار بودن یا نبودن تأثیری برای ما نداشته است . (حالت بازی)

در هر بار حلقه‌ی while اصلی که بررسی خالی نبودن صف را انجام می دهیم می آییم و برای هر رأس همسایه‌ی min گرفته شده (که یال های جهت دار خود به خود در اینجا تأثیر خود را گذاشته اند ؛ یعنی اگر یال $u \to v$ باشد ، اما $u \notin adj\, v$ ، $v \in adj\, u$) . روی یال relax می کنیم .

II) می دانیم که ریلکس کردن صرفاً key v را تغییر می دهد درصورت لزوم . پس مجدداً جهت دار بودن گراف تأثیری روی این مرحله هم ندارد .

III) از آنجایی که این آخرین مرحله‌ی کار بود و خروجی آن پس از حلقه مشخص می شود و اینکه بررسی کردیم جهت دار بودن گراف تأثیری روی کار ندارد ، پس dijkstra روی گراف های جهت دار با وزن منفی درست کار می کند . (نتیجه گیری)

نبود

(نوع خروجی)

IV) تنها یک نکته باقی می ماند . از آنجایی که خروجی dijkstra یک MST است و در MST هیچ دوری موجود نمی باشد ؛ اگر مسیر $u \to s$ وجود داشته باشد shortest path هم باشد آنگاه مسیر $s \to u$ اصلاً وجود ندارد . زیرا گراف ما جهت دار می باشد و برای وجود مسیر بازگشتی از هر رأسی به src باید در داشته باشیم که ممکن نیست .

⟹ پس dijkstra فقط برای خروجی single source درست کار می کند . (در گراف های جهت دار)

سوال ۵ : ب)

چون مسیر مطمئن شامل یال هایی با مقدار نزدیک به ۱ می باشد ؛ در این قسمت یک تغییر کوچک روی الگوریتم dijkstra می دهیم تا مسیری مطمئن را پیدا کند .

— (می دانیم که هر یال وزنی بین ۰ و ۱ دارد پس یال منفی یا دور منفی نداریم)

— (از روی مجموعه V ها می توان مسیر را یافت)

— (اگر $d[v]$ یک راس ۰ باشد یعنی مسیر مطمئن وجود ندارد)

— (اگر $d[v] = \infty$ یعنی مسیری وجود ندارد)

Algorithem :

New method

— Rush (u, v)

if $d[v] < d[u] \times w(u,v)$

$d[v] = d[u] \times w(u,v)$

$\pi[v] = u$

— Dijkstra (root)

for $v \in V$

$d[v] = -\infty$

$\pi[v] = nil$

$d[root] = 1$

$Q \leftarrow V \longrightarrow$ priority Queue Based on keys (d)
(Max Queue)

while Q != Null

$u \leftarrow$ extract_Max ()

for each $v \in adj[u]$

Rush (u, v)

← بعد از آن از روی رأسی که می خواهیم تا زمانی که به root نرسیده ام به $\pi[v]$ می رویم .