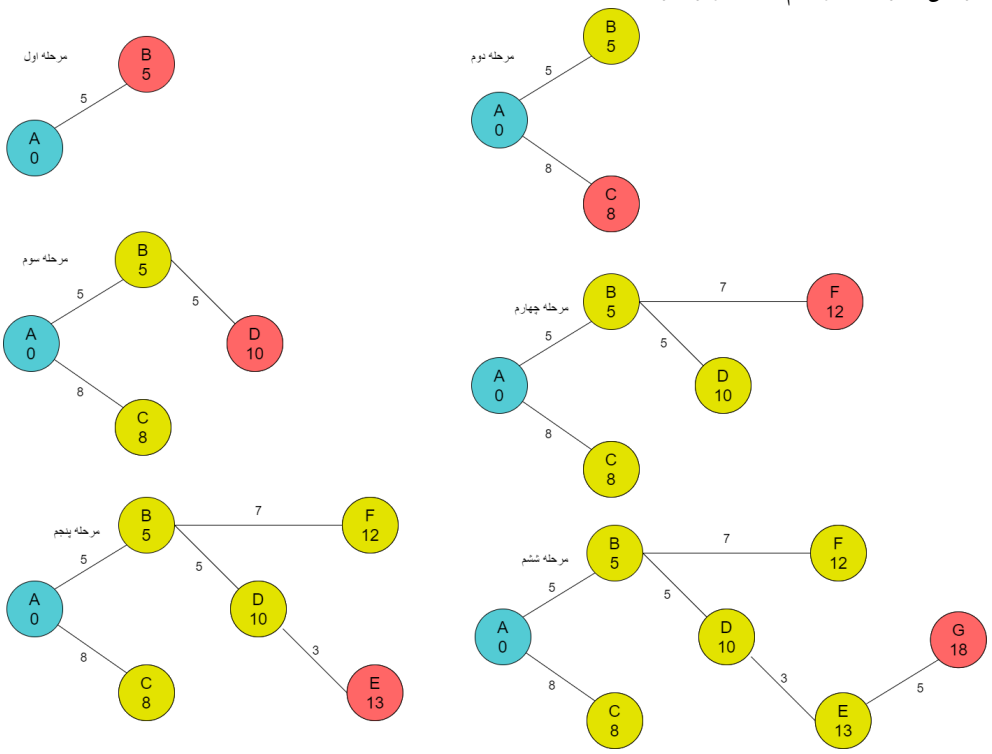
# دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر
# تحلیل و طراحی الگوریتم ها

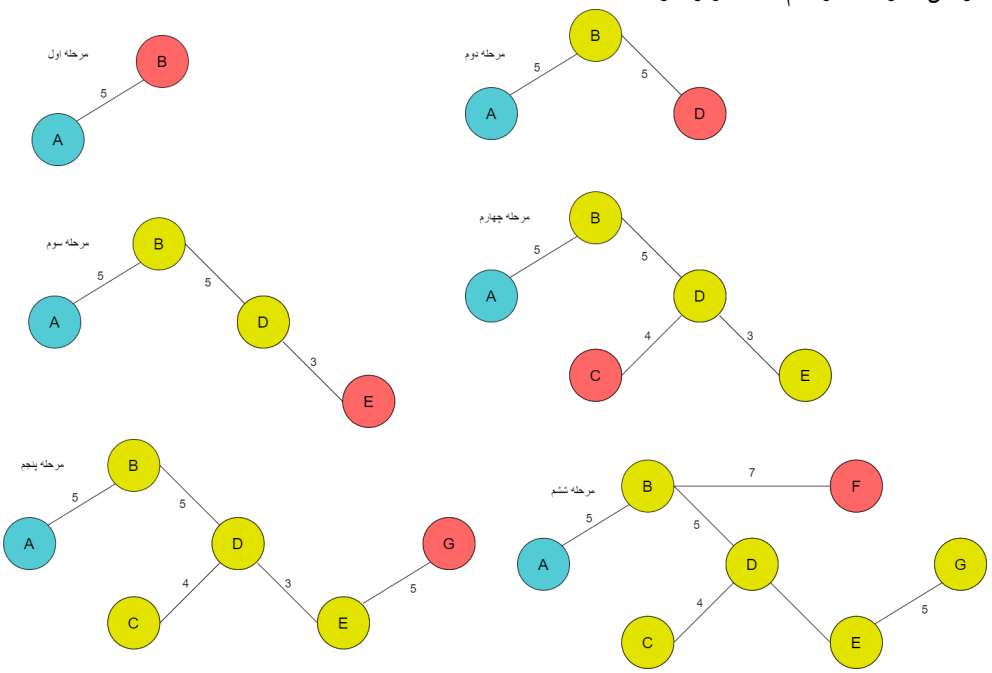
پاسخ تمرین کتبی چهارم
طراح: امیرحسین علیزاد، aalizad79@gmail.com

۱. باشماره دانشجویی ۸۱۰۱۹۷۵۴۶ به این سوال پاسخ میدهیم.

(آ) مراحل اجرای الگوریتم در تصویر زیر آمده است.




(ب) مراحل اجرای الگوریتم در تصویر زیر آمده است.



(ج) همانگونه که مشاهده کردید درخت حاصل از اجرای دو الگوریتم فوق با یکدیگر متفاوت است. این دو الگوریتم در اجرا شباهت زیادی به یک دیگر دارند اما درخت تولید شده از آن‌ها دارای خواص متفاوتی است. درخت کوتاه ترین فاصله، کوتاه ترین فاصله هر گره از گره مبدا را مشخص می‌کند و در هر مرحله راس با کمترین فاصله پیدا شده به مجموعه اضافه می شود. از طرف دیگر، درخت پوشای کمینه، کم وزن ترین درختی است که همه گره‌های گراف را به هم وصل می‌نماید و لزوما شامل کوتاه ترین فواصل نیست و در هر مرحله از بین یال های ممکن یال با کمترین وزن را انتخاب کرده و به مجموعه اضافه می کند.

۲. چون فقط هزینه یکی از یال‌ها کاهش یافته است، برای محاسبه فاصله جدید کافی است که برای هر دو گره دلخواه فاصله آن‌ها تا دو سر یال uv محاسبه کنیم و با وزن کنونی آن‌ها مقایسه کنیم. هرکدام که کوچکتر بود به عنوان مسیر کمینه بین دو گره معرفی می‌کنیم. به این ترتیب داریم:

```
for i in 0...N-1
    for j in 0...N-1
        d[i][j] = min(d[i][j],d[i][u]+w(uv)+d[v][j])
        d[i][j] = min(d[i][j],d[j][u]+w(uv)+d[v][i])
```

هزینه این الگوریتم $O(n^2)$ است.

اثبات: برای دو راس دلخواه i و j:

(آ) اگر کوتاه ترین مسیر بین این دو راس دلخواه از یال uv عبور نکند چون فقط وزن همین یال عوض شده است پس مقدار کوتاه‌ترین مسیر با مقدار در گراف قبلی برابر است.
حال باید ثابت کنیم مسیر $i \to uv \to j$ یا عکس آن مقدار کمتری از کوتاه ترین مسیر ندارد. اگر مقدار عبارت $d[i][u] + w(uv) + d[v][j]$ برقرار باشد، در این صورت مسیر $i \to uv \to j$ کوتاه‌ترین خواهد بود که با فرض تناقض دارد که کوتاه‌تری مسیر شامل uv نیست.

(ب) اگر کوتاه ترین مسیر بین این دو راس دلخواه از یال uv عبور کند، با فرض عدم وجود دور منفی خواهیم داشت که مسیرهای $i \to u$ و $v \to j$ مسیر ساده هستند و دوری ندارند. دو مسیر $i \to u$ و $v \to j$ کوتاه‌ترین مسیر بین جفت گره‌های نام برده هستند. این موضوع را فقط برای $i \to u$ ثابت می‌کنیم و برای دیگری نیز مشابه است. فرض کنید مسیر دیگری مثل P' باشد که از مسیر $i \to u$ که آن را P می‌نامیم و در گراف اولیه است کوتاه‌تر باشد. P' از uv عبور نمی‌کند چون اگر می‌کرد می‌شد با عدم عبور از آن اندازه مسیر را کوتاه تر کرد. بنابراین اندازه مسیر P و P' برابر است.
حال کافی است که نشان دهیم که $d[i][j] > d[i][u] + W(uv) + d[v][j]$ است. d[i][j] مقدار فاصله کمینه این دو راس در گراف قبلی است. از طرفی ثابت کردیم که مقدار فاصله $i \to u$ و $v \to j$ در دو گراف برابر و کمینه است، پس تنها تفاوت دو مسیر اندازه یال uv است که در گراف دوم کمتر است پس این مسیر کمینه است.

۳. یک گراف جهت دار تشکیل میدهیم که حروف راس های آن هستن و اگر بتوان حرف x را به حرف y با هزینه c تبدیل کرد یک یال جهت دار از راس x به y با وزن c می کشیم. حال برای تمامی جفت حروف الفبا کوتاه ترین مسیر بین آن ها را پیدا میکنیم. برای یکسان کردن کاراکتر های p و q ، ۳ حالت داریم:

آ. کاراکتر p به q تبدیل شود.
ب. کاراکتر q به p تبدیل شود.
ج. هر دو کاراکتر p و q به کاراکتر سومی مانند z تبدیل شوند.

برای حالت سوم باید روی ماتریس کوتاه ترین مسیر پیمایشی انجام دهیم. هزینه زمانی این الگوریتم برابر یک بار اجرای الگوریتم فلوید وارشال و در نهایت پیمایش روی ماتریس کوتاه ترین مسیر به ازای هر کاراکتر موجود در رشته ها است و برابر $O(26^3 + n * 26)$ می باشد.(۲۶ برابر تعداد حروف الفبا می باشد و در صورتی که پاسخ شما تنها بر حسب n باشد نیز صحیح است)

۴. ابتدا با استفاده از الگوریتم فلوید وارشال کوتاه ترین مسیر میان تمام زوج راس ها را محاسبه می کنیم. حال با استفاده از تعریف یال جذاب خواهیم داشته که برای تمام یال های موجود در گراف به صورت $e = (a, b)$ ، اگر $D[u, a] + w_e + D[b, v] \leq L$ یال e جذاب خواهد بود. هزینه اجرای این الگوریتم برابر یک بار اجرای الگوریتم فلوید وارشال و یک بار پیمایش روی تمامی یال ها است و برابر $O(n^3 + |E|)$ است.

۵. در این سوال باید سعی کنید دنباله ای را با توجه به محدودیت هایی که دارد تولید کنید. صورت عبارات را بازنویسی می کنیم.( دنباله ها را با شروع از ۰ در نظر میگیریم و $S_i$ را باز تعریف میکنیم یعنی $S_i = \sum_{j=0}^{i-1} a_j$ که مثلا $S_n$ برابر مجموع کل آرایه شود). پس شرط a - b > c به $S_{b+1} - S_a > c$ و شرط a - b < c به $S_{b+1} - S_a < c$ تبدیل می شود. نا مساوی های به صورت $S_{b+1} - S_a > c$ را به صورت $S_a - S_{b+1} < -c$ تبدیل می کنیم. در یک گراف فرضی که n راس دارد و هر راس از ۱ تا n نام گذاری شده اند به ازای هر کدام از این قوانین یالی بکشیم (از راس a به راس b با وزن c - 1 )و کوتاه ترین مسیر ها را با استفاده از آن پیدا کنیم این $S_i$ ها که متناظر با $dist[0][i]$ هستند تمام شرایط سوال را دارند بجز زمانی که در این گراف دور منفی وجود داشته باشد. پس می توان با ساختن این گراف و با استفاده از الگوریتم بلمن فورد $S_i$ ها را پیدا کرد و هزینه زمانی آن نیز O(n(n + m)) می باشد.

۶. الگوریتم کساراجو را بررسی می کنیم:

با استفاده از الگوریتم DFS و استفاده از یک stack و در اجرای الگوریتم DFS پس از اینکه تمام فرزندان راس v پیمایش شدند راس v را در stack وارد کرده و پس از اجرای الگوریتم تمامی یال های گراف را برعکس ( transpose میکنیم. حال تا زمانی که stack خالی نشده باشد عنصر سر stack را pop کرده و از راس pop شده الگوریتم DFS را اجرا میکنیم.(روی گراف جدیدی که با برعکس کردن یال های گراف اصلی بدست آورده ایم) با این کار از راسی که الگوریتم DFS را شروع میکنیم مولفه ی قویا همبندی که راس مورد نظر شامل آن است را به ما میدهد. اهمیت عملیات DFS اول بر این است که ما با استفاده از این کار میتوانیم راس مناسبی برای شروع عملیات ساختن مولفه های همبندی پیدا کنیم. در صورت شروع از راس نامناسب، الگوریتم DFS به ما درختی واحد می دهد که مولفه های قویاً همبند آن از هم جدا پذیر نیستند.