## نمونه سوالات بخش ۸,۱

**۱,۸,۱ فرض کنید گراف G داده شده است به صورتی که وزن هر یال یا یک است یا m. الگوریتمی با مرتبه زمانی O(V+E) بنویسید که کوتاه ترین مسیر از یک راس S به راس t در گراف ما را مشخص کند.**

**راه حل:**

**از گره s جستجوی سطح اول میزنیم. دو صف برای گره های اضافه شده با فاصله یک و دیگری برای گره های اضافه شده با فاصله M در نظر میگیریم. برای انتخاب گره بعدی در جستجوی سطح اول با توجه به فاصله های ثبت شده در ابتدای دو صف، گره بعدی را انتخاب میکنیم. فاصله نهایی گره های خارج شده از صف دوم را در صورتی که قبلا از صف خارج نشده بودند در نظر میگیریم و اگر قبل از خارج شدن آنرا با استفاده از صف یک پیموده بودیم دیگر آن گره را در نظر نمیگیریم.**

---

**۲,۸,۱ فرض کنید گراف G داده شده است به صورتی که وزن یال های آن عددی بین صفر و N است. الگوریتمی ارایه دهید که کوتاه ترین مسیر از یک راس s به t را با زمان O(E + V*N) و حافظه O(V + N) مشخص کند.**

**راه حل:**

**یک آرایه از صف ها داریم، از گره s جستجوی سطح اول میزنیم. گره های مجاور را با توجه به فاصله شان در صف مربوطه به خود اضافه میکنیم. برای مثال اگر گره فعلی فاصله a از s داشته باشد و یال به گره مجاورش وزن b داشت، گره مجاور را در صف a+b اضافه میکنیم. توجه کنید که صف ها در خود فاصله ها را نیز نگه میدارند پس لازم نیست دفعه ی بعدی صف ها با فاصله کمتر از a را بررسی کنیم. فاصله های نهایی گره ها را هنگام خارج شدن از صف مقدار دهی میکنیم و اگر گره ای را قبل از خارج شدن پیموده بودیم، دیگر در نظر نمیگیریم.**

---

**۳,۸,۱ گراف دو بخشی گرافی است که بتوان رئوس آنرا به دو دسته افراز کرد به نحوی که در بین رئوس دسته ها یالی وجود نداشته باشد. الگوریتمی ارایه دهید که با گرفتن اطلاعات یک گراف به ما بگوید دوبخشی است یا نه.**

**راه حل:**

**با یک بار BFS زدن روی گره ها حرکت میکنیم و گره ها به دو بخش تقسیم میکنیم. گره نخست را در یکی از بخش ها قرار میدهیم و به ترتیب گره هایی را که میبینیم در بخش مخالف قرار میدهیم. در حین حرکت اگر به تناقضی برخوردیم یعنی گراف دو بخشی نیست و در غیر این صورت گراف دو بخشی است زیرا به بخش مجزا افراز شده است.**

---

**۴،۸،۱ در یک درخت، قطر را چگونه میتوان پیدا کرد؟**

**راه حل:**

**از یکی از گره ها شروع میکنیم و BFS میزنیم و دور ترین گره به این گره را بدست میاوریم. از گره جدید دوباره BFS میزنیم و دور ترین گره را به این گره نیز به دست میاوریم. مسیر به دست آمده بین این دو گره قطر درخت است.**

---

**۵،۸،۱ یک نینجا در ته چاهی با دو دیوار که هر کدام از n بلوک آجری تشکیل شده است گرفتار شده. ویژگی این چاه ها در این است که بعضی از بلوکه های دیواره های این چاه سمی است. نینجه ما در ابتدا در پایین ترین نقطه این چاه است. او در هر مرحله میتواند یک بلوک بالا یا پایین برود یا به k بلوک بالاتر در دیوار مخالف بپرد. با هر حرکتی که نینجا ما میکند سطح آب کف چاه یک بلوک بالاتر میاید. در این شرایط که به دیوار های سمی هم نباید بخورد الگوریتمی با گرفتن اطلاعات دیوار ها عدد n ارایه دهید تا نینجا نجات پیدا کند.**

**راه حل:**

**بلوک ها را گره گراف در نظر میگیریم و هر گره به گره بالایی و پایینی و n تا بالاتر از خودش یال دارد. گره های سمی و یال هایشان را حذف میکنیم و از دو گره پایینی BFS میزنیم و در حین حرکت بررسی میکنیم که سطح گره از سطح فعلی پایین تر نباشد.**

---

**۶،۸،۱ در مسائل مربوط به پیدا کردن کوتاه ترین مسیر بین دو راس از یک گراف اغلب با تعداد بیشتر از یک مسیر مواجه هستیم، الگوریتمی با پیچیدگی زمانی خطی بدست آورید که تعداد تمام کوتاهترین مسیر های متمایز بین دو راس u و v را پیدا کند.**

**راه حل:**

**با تغییر دادن BFS الگوریتمی برای حل این مسئله می یابیم. از راس v الگوریتم را شروع می کنیم در طی اجرای الگوریتم ما همواره دو متغیر اضافی برای هر راس ذخیره می کنیم، یکی distance که فاصله این راس تا راس v است و دیگری تعداد مسیر ها با طول distance از راس v. الگوریتم بر این اساس کار می کند: هر مسیر به طول k از u به v را می توان به مسیری به طول k-1 از u به همسایه های v و مسیری به طول یک به راس v تقسیم کرد. بنابراین اگر ما تعداد کوتاه ترین مسیر ها به تمامی راس های میانی رو دنبال کنیم به تعداد تمام کوتاهترین مسیر ها بین u و v میرسیم.**

```
function NumShortestPaths(G,u,v)
Let Q be an empty queue
u.visited = true
u.distance = 0
u.numPaths = 1
add u to Q
while Q is not empty do
curr = Q.dequeue()
for each neighbor t of curr do
if t.visited == false then
t.visited = true
t.distance = curr.distance + 1
t.numPaths = curr.numPaths
add t to Q
else
if t.distance == curr.distance + 1 then
t.numPaths += curr.numPaths
end if
end for
end while
return v.numPaths
```

**۷،۸،۱ فرض کنید بخواهیم یک گراف غیرجهتدار و همبند را با دو رنگ سیاه و سفید رنگ آمیزی کنیم به طوری که راس های مجاور همرنگ نباشند. اثبات کنید اگر گراف دوری به طول فرد داشته باشد همچین رنگ آمیزی ای وجود ندارد.**

**راه حل:**

**برهان خلف. فرض می کنیم این کار درحالی که گراف دور فرد دارد شدنی باشد.فرض میکنیم v1 سفید، v2 سیاه، v3 سفید و به همین ترتیب راس ها با ایندکس فرد سفید و راس های با ایندکس زوج سفید باشند. از انجایی که اندیس راس آخر فرد است پس سفید است و این راس با راس ۱ مجاور است، بنابراین هر دو همرنگ خواهند بود پس برهان خلف باطل می باشد و نمی توانیم این رنگ آمیزی را داشته باشیم.**

**الگوریتمی با پیچیدگی O(v+e) بیابید که این رنگ آمیزی را با در نظر گرفتن این نکته که گراف دور فرد نداشته باشد، بیابد.**

**فرض کنید راس v سفید است، روی این راس الگوریتم BFS را اجرای می کنیم و تمام راس های level اول باید سیاه باشند چون راس اول سفید بود، سپس تمام راس های level دوم باید سیاه باشند و به همین ترتیب شما می توانید با استفاده از پارامتر فاصله ی هر راس با راس v، تمام راس های گراف را رنگ آمیزی می کنیم، پیچیدگی این الگوریتم همان پیچیدگی BFS است یعنی همان O(v+e).**

---

**۸،۸،۱ در یک کشور ناکجاآباد n شهر وجود دارد که توسط n-1 جاده به هم متصل شده اند؛ در این کشور از هر شهر میتوان به شهر دیگر رفت. بر اثر سیلُ تعدادی از جاده ها خراب شده است و نیاز به تعمیر دارد. برای تعمیر هر جادهُ اداره راه شهر های دو سمت این جاده باید به صورت همزمان مشغول به تعمیر جاده شوند. تعمیر هر جاده دقیقا یک روز زمان میبرد. همچنین اداره راه هر شهر در یک روز تنها میتواند روی یک جاده کار کند. تعداد کمترین روز هایی که برای تعمیر همه جاده های تخریب شده نیاز است چقدر است؟ یک برنامه زمانی پیشنهاد دهید که نشان در هر روز کدام جاده ها باید تعمیر شوند.**

**راه حل:**

**از شهری که بیشترین جاده خراب k را دارد BFS و هر جاده تخریب شده را با یک رنگ از k رنگ، رنگ آمیزی میکنیم به طوری که جاده های مجاور همرنگ نباشند. جاده های همرنگ را در یک روز تعمیر میکنیم.**

---

**۹,۸,۱ دانشجویان کامپیوتر همیشه بعد از کلاس ساختمان داده با هم این بازی را انجام میدهند. ابتدا در یک صف به ترتیب قد قرار میگیرند (هیچ دو نفر قد یکسان ندارند). سپس در هر مرحله از بازی هرکسی با احتمالی خاص کوتاه ترین فرد را از بازی حذف میکند اما خودش را نمیتواند حذف کند. شما باید با دانستن مقدار احتمال برای دانشجویان بگویید که پس از حداکثر k مرحله چه زیر مجموعه ای از دانشجویانی که در ابتدا شروع به بازی کرده اند ممکن است باقی بمانند. (دقت کنید که احتمال دو نفر متفاوت لزوما متفاوت و یکسان نیست و این احتمال میتواند هر مقداری میان بازه بسته صفر و یک باشد)**

**راه حل:**

**هر حالتی که ممکن است بوجود بیاید شرایط زیر را دارد:**

**M -> مرده**

**Z -> زنده**

**M* -> تعداد بزرگتر مساوی صفر مرده**

**Z* -> تعداد بزرگتر مساوی صفر زنده**

**پس هر حالت را میتوان با دو زنده ابتداییش به صورت یکتا مشخص کرد و آنرا با (A, B) نشان میدهیم.**

**از هر حالت به چهار حالت میتوان برویم:**

**(A, B) -> (A, B)**

**به شرط آنکه احتمال هیچ کس ٪۱۰۰ نباشد**

**(A, B) -> (A, B+1)**

**به شرط آنکه A صفر درصد و از B به بعد کسی ٪۱۰۰ نباشد**

**(A, B) -> (B, B+1)**

**به شرط آنکه A ٪۱۰۰ نباشد و از B به بعد همه صفر درصد نباشد**

**(A, B) -> (B+1, B+2)**

**به شرط آنکه A صفر درصد نباشد و از B به بعد همه صفر درصد نباشند**

**چون تعداد حالت ها محدود است میتوان از حالت آغازین به عمق k، BFS زد.**

**۱۰،۸،۱ معین میخواهد شهر را بگردد. برای اینکار او میخواهد حداقل یکبار همه خیابان های شهر را ببیند. شما باید به او کمک کنید تا هرچه سریعتر بتواند این کار را انجام دهد. او میداند که طول همه خیابان های شهر عددی بین یک تا w است و به همه تقاطع ها دقیقا ۴ خیابان متصل است به جز دو تقاطع که سه خیابان دارند. او میخواهد از جایی شروع به حرکت کند و همه خیابان هارا ببیند و به نقطه شروع بازگردد و این کار را در سریع ترین زمان ممکن انجام دهد.**

**راه حل:**

**مساله مدل میشود به گرافی همبند با دو راس درجه فرد و تعدادی راس با درجه زوج. انتخاب دنباله ای از یال ها به طوری که هر دو یال مجاور در دنباله با هم در یک راس اشتراک داشته باشند و همه یال ها در دنباله حداقل یکبار آمده باشد.**

**مسیری که انتخاب میشود درواقع دوری اویلری در گراف است که از بعضی از یال ها بیش از یکبار استفاده شده است. اگر یال هایی که یکبار استفاده شده اند را کنار بگذاریم باقی یال ها تبدیل میشوند به مسیری از یکی از دو راس فرد به دیگری. پس مساله تبدیل میشود به پیداکردن کوتاه ترین مسیر بین این دو راس و اضافه کردن آن به گراف تا تبدیل به گراف اویلری شود.**

**برای اینکار آرایه ای از صف هارا در نظر میگریم از یکی از گره ها BFS میزنیم. گره های مجاور را با توجه به فاصله شان در صف مربوط اضافه میکنیم. برای مثال اگر گره فعلی فاصله a از راس ابتدایی داشت و یال به گره مجاورش وزن b را داشت گره مجاور را در صف a+b اضافه میکنیم. به این ترتیب هر گره در صف با شماره فاصله اش اضافه شده است. فاصله ها را در هنگام خارج شدن از صف نهایی میکنیم و اگر گره ای قبلا پیموده شده بود دیگر آنرا در نظر نمیگیریم و کوتاه ترین فاصله بین دو راس با درجه فرد به دست میاید.**

---

**۱۱،۸،۱ برخی از کاربرد های جست و جوی اول-سطح را نام ببرید.**

**راه حل:**

**مسئله ی کوتاه ترین مسیر و درخت پوشای کمینه برای گراف های بدون وزن: در گراف های بدون وزن کوتاه ترین مسیر، مسیری است که شامل کمترین یال ها باشد، با BFS زدن روی یک راس همیشه با کمترین تعداد یال ها به مقصد می رسیم، همچین در گراف های بدون وزن هر درختی پوشایی، درخت پوشای کمینه است بنابراین با استفاده از BFS می توانیم همچین درختی ایجاد کنیم.**

**سیستم های مسیر یابی: BFS تمام موقعیت های همسایه را به ما می دهد.**

**زباله روبی در زبان های برنامه نویسی مانند جاوا: با استفاده از BFS حافظه ای که دیگر مورد استفاده ی برنامه نیست شناسایی میشود و حذف میشود.**

**پیدا کردن دور در گراف های غیر جهت دار: در گراف های غیر جهت دار می توان با این الگوریتم دور را شناسایی کرد در حالی که در گراف های جهت دار این کار با BFS امکان پذیر نیست.**

**تشخیص دو بخشی بودن یا نبودن گراف: با استفاده از BFS و رنگ آمیزی یکی در میان هر level این کار امکان پذیر است.**

**پیدا کردن مسیر بین دو راس: با استفاده از این الگوریتم می توان مسیر بین دو راس را در صورت وجود یافت.**

**پیدا کردن تمام راس های یک مولفه ی همبند: با استفاده از BFS می توان تمام راس های قابل دسترس از راس داده شده را یافت.**

**همچنین بسیاری از الگوریتم های معروف همچون الگوریتم پرایم برای یافتن درخت پوشای کمینه یا الگوریتم داکسترا برای یافتن کوتاه ترین مسیر با مبدا معین از ساختاری مشابه BFS استفاده می کنند.**

---

**۱۲،۸،۱ چه زمانی بهتر است از الگوریتم های BFS یا DFS استفاده کنیم؟**

**راه حل:**

**اینکه برای حل مسئله از کدام روش جست و جو استفاده کنیم تا حدود زیادی به مسئله و ساختار درختی که می خواهیم جست و جویش کنیم و همچنین شرایط و ویژگی های آن بستگی دارد. اگر بدانیم پاسخ از ریشه ی زیاد فاصله ای ندارد، BFS بهتر خواهد بود. اگر درخت ما عمیق باشد و پاسخ ها مسئله کمیاب باشند DFS به احتمال زیاد زمان بسیار زیادی طول خواهد کشید، در عوض BFS سریع تر خواهد بود. اگر درخت ما دارای پهنای زیادی باشد و عریض باشد BFS نیاز به حافظه ی بسیار زیادی خواهد داشت، بنابراین نشدنی خواهد بود. اگر جواب در درخت ما زیاد رخ دهد و در عمق باشد BFS ممکن است نشدنی باشد، اگر درخت بسیار عمیق باشد میبایست در جست و جوی DFS برای پیمودن عمق درخت محدودیتی قائل شوید وگرنه اینکار بسیار طولانی و غیر ممکن خواهد شد.**

---

**۱۳،۸،۱ وقتی از ماتریس مجاورت استفاده می کنیم، اکثر الگوریتم های گراف با پیچیدگی زمانی امگا V*V کار می کنند، ولی چندین استثنا نیز وجود دارد. نشان دهید با داشتن ماتریس مجاورت گراف، تشخیص اینکه آیا گراف ما دارای یک "حفره ی فراگیر"(راسی با درجه ی ورودی v - 1 و درجه ی خروجی ۰ ) است در زمان O(v) انجام می شود.**

**راه حل:**

**ابتدا توجه به این دو نکته ضروری است اول اینکه در ماتریس مجاورت، عناصر ردیف i ام، یال های خارج شده از راس i به بقیه راس ها را نشان میدهد. و عناصر ستون j ام، یال های وارد شده به راس j را نشان میدهد. و دوم اینکه از آنجایی که از "حفره ی فراگیر" به بقیه ی راس ها یالی خارج نمی شود بنابراین ماکسیمم ۱ همچین راسی وجود خواهد داشت.**

**ما به دنبال حفره ی فراگیر هستیم، راسی با درجه ی ورودی v - 1 و درجه ی خروجی ۰. کاری که ما باید انجام بدیم در واقع پیدا کردن k است، به طوری که ردیف k ام تماماً صفر و ستون k ام تماماً ۱ به جز عنصر k ام که باید صفر باشد. از پیمایش ردیف اول شروع می کنیم، و به محض مشاهده ی اولین ۱ متوقف میشویم. این کار در بدترین حالت O(V) طول میکشد. اگر به ۱ بر نخوردیم و از انجای که حفره ی فراگیر یکتاست پس یعنی این ردیف تنها کاندید احتمالی برای حفره ی فراگیر است. سپس بر روی اولین ستون در زمان O(V) بررسی می کنیم که آیا تمام عناصر ان ها ۱ میباشد یا نه در صورتی که اینگونه باشد راس اول "حفره ی فراگیر" است، در غیر اینصورت این گراف همچین راسی ندارد. الگوریتم زمانی پایان می پذیرد که ما به ردیفی با تمام عناصر صفر بر بخوریم و بررسی کنیم که آیا این راس "حفره ی فراگیر" هست یا خیر. بنابراین مطمئن هستیم که هزینه ی این الگوریتم O(V) باقی می ماند.**

---

**۱۴،۸،۱ الگوریتم BFS را به گونه ای تغییر دهید که بتواند کوتاه ترین مسیر در گراف وزن دار را بدست بیاورد.**

**راه حل:**

**برای این منظور ابتدا گراف وزن دار را به گراف غیر وزن دار تبدیل می کنیم به طوری که به ازای هر یال با وزن x به تعداد x - 1 راس بین این دو راس قرار می دهیم و در نتیجه ۱ یال به x یال متوالی تبدلی میشود. حال اگر بر روی این گراف جدید الگوریتم BFS را اجرا کنیم، کوتاه ترین مسیر ها را به ما می دهد. از آنجایی که تعداد یال ها و راس های گراف جدید هم order با تعداد یال ها و راس های گراف اصلی است بنابراین این الگوریتم نیز با پیچیدگی o(v+e) کوتاه ترین مسیر ها را به ما می دهد.**