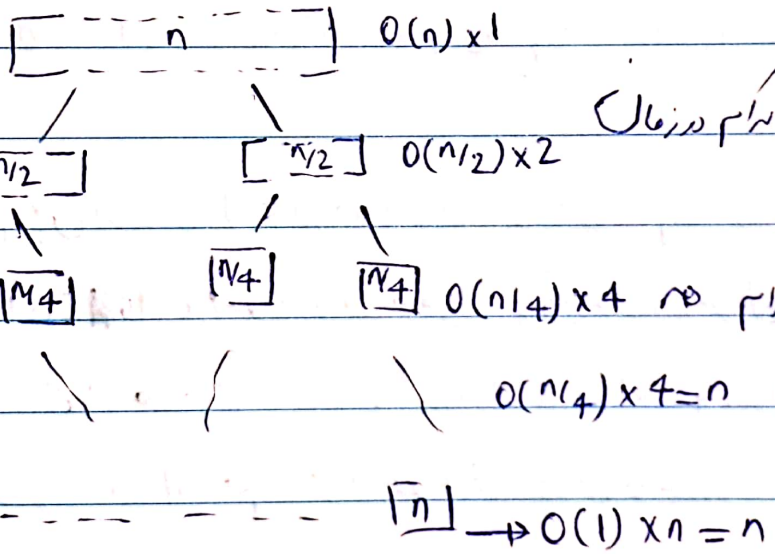
۱) ابتدا با استفاده از الگوریتم Quickselect، هر بار میانه را پیدا کرده و به عنوان pivot در نظر میگیریم. الگوریتم Quickselect این کار را در زمان $O(n)$ انجام می‌دهد. حال چون هر بار بعد از تقسیم آرایه این کار برای آرایه انجام شود. پس پیچیدگی زمانی پیدا کردن میانه در همه‌ی مراحل Quicksort به شکل زیر است.



۲ آرایه به طول $n/2$ داریم که میانه هر کدام در زمان $O(n)$ پیدا می‌شود پس $O(n/2)\times 2 = O(n)$ میانه ۲ آرایه

۴ آرایه داریم که به طول $n/4$ به هر کدام $O(n/4)\times 4$ میانه

طول زمان $O(n/4)$ پیدا می‌شود پس $O(n/4)\times 4 = n$

$O(1)\times n = n$

$= O(n \log n)$

$O(n) \times O(\log n)$ یعنی به تعداد دفعات تقسیم برابر ($O(\log n)$ یا همان ارتفاع درخت) این کار انجام می‌شود. در نتیجه

همان چون در هر مرحله میانه را برداشته‌ایم پس حالت بهینه Quicksort اتفاق می‌افتد یعنی

$O(n \log n) + O(n \log n) =$ ← در نتیجه با در نظر گرفتن پیچیدگی زمانی پیدا کردن میانه داریم $O(n \log n)$

$2O(n \log n) = O(n \log n)$

2) ابتدا در Dag ، مرتب سازی موضعی اجرا می‌دهیم (یا همان Topological Sort) این کار در زمان $O(|V|+|E|)$ اجرا می‌شود.

می‌توان از DFS برای مرتب آوردن زمان پایان گره‌ها در مراحل استفاده کرد که در زمان $O(|V|+|E|)$ اجرا می‌شود. مسیر پوشا فقط در صورتی وجود دارد که بین هر دو راس متوالی در ترتیب بدست آمده، یالی وجود داشته باشد.

$u \to w \to v \to e$

اگر شرط بالا برقرار باشد، ترتیب به دست آمده از Topological Sort، مسیر مطلوب خواهد بود.

وجود چنین یالی را می‌توان در زمان $O(n)$ دارد.

$O(|V||E|)$.

3) با الگوریتم بلمن فورد می توان دور منفی را پیدا کرد. می دانیم در الگوریتم بلمن فورد، پیچیدگی زمانی برابر با یا در اصل $O((|V|-1)|E|)$ است. و در هر مرحله با توجه به وزن هر یال و دو راس شامل آن یال، این مقادیر به روز می شوند. حال اگر دور منفی نداشته باشیم. در مرحله ی آخر الگوریتم باید آخرین تغییرات اعمال شده باشد و در واقع اگر یک بار دیگر (بار $|V|$ام) بخواهیم وزن رئوس را به روز کنیم هیچ تغییری نباید ببینیم.

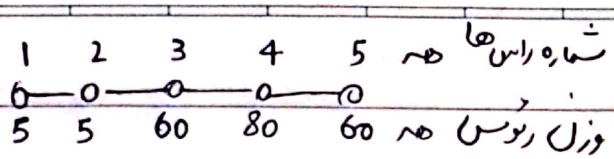
بعد از
7

اما اگر در مرحله‌ی آخر وزن راسی تغییر کند (کمتر شود) به این معنی است که دور منفی داریم زیرا با پیمودن دور منفی می‌توان مسیر با وزن کوتاه‌تری پیدا کرد یعنی اگر دوباره یعنی می‌توان به تعداد بی‌شماری دور منفی را طی کرد و به مسیری کوتاه‌تر دست یافت. پس فقط کافیست یکبار دیگر الگوریتم بلمن فورد را روی گراف پیاده کنیم.

سعید سرادی 9613028

(A)

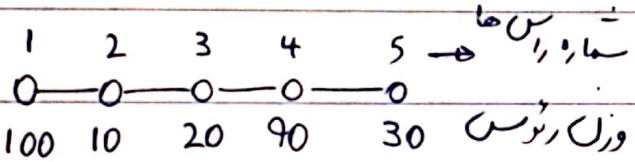
4) مثال نقض ←



هستند

این الگوریتم راس های 4 و 2 (یا 4 و 1) انتخاب می‌شوند درصورتی که جواب صحیح راس های 1 و 3 و 5

(B) مثال نقض ←



این الگوریتم راس های 1 و 3 و 5 انتخاب می‌شوند حال $w(1)+w(3)+w(5) > w(2)+w(4)$

درصورتی که جواب هفت راس های 1 و 4 هستند

الگوریتم مناسب: ابتدا راس ها را از v_1 تا v_n نامگذاری می‌کنیم. حال برای هر راس v_i مقدار $d(v_i)$ را برابر با اصلاح مجموع وزن های همسایه از وزن خود راس تعریف می‌کنیم یعنی $d(v_i) = w(v_i) - w(v_{i-1})$ (راس)

حال بین اینها مینیمم را انتخاب می‌کنیم و به مجموعه مستقل اضافه می‌کنیم. حال $d(v_i)$ را برای همسایه(ها)ی راس انتخاب شده به روز می‌کنیم. سپس همین روند را تکرار می‌کنیم. ناهمواری راس ها زمانی از وضعیت های "حذف شده" یا "انتخاب شده" خارج نمی‌شود.