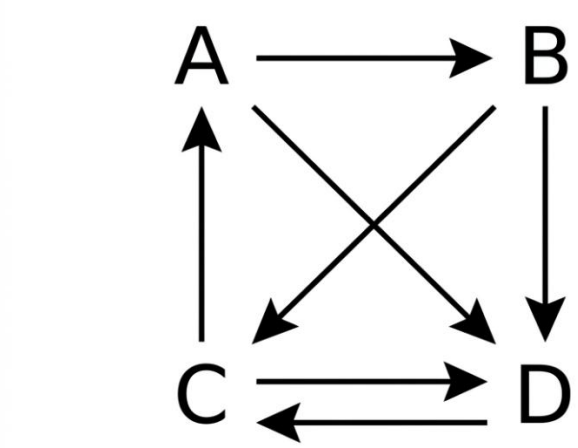
Data Structures and algorithms
Instructor : **Dr.Amin Eskandari**
Head-Ta's : Hamid Namjoo , Amir Hossein Hemati , Ali Mojahed
Ta's : Amir Mohammad Asadjoo , Padina razmjooi , Armin Janfaza , Alireza Heydari , Mahdi Torabi ,
Maryam Ghorbani , Fatemeh Keshavarz , MohammadReza Fasli , Abolfazl Tadrisi

Week 08 Answers

پاسخنامه تکالیف هفته 08

پاسخ سوال اول:

الف)



ب)

	A	B	C	D
A	0	1	0	1
B	0	0	1	0
C	1	0	0	1
D	0	0	1	0

پاسخ سوال دوم :

a)

	1	2	3	4	5
1	0	0	1	0	0
2	1	0	0	1	0
3	0	0	0	0	0
4	0	0	1	0	1
5	1	0	0	0	0

b)

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	0	0
4	0	0	1	0	0
5	0	0	1	0	0

c)

	1	2	3	4	5
1	0	1	0	1	0
2	0	0	0	1	0
3	1	0	0	1	0
4	0	0	0	0	1
5	0	0	0	0	0

پاسخ سوال سوم:

فایل A3.py را اجرا کنید

ایده حل

این مسئله دقیقاً کاربرد کلاسیک BFS است.

در گراف بدون وزن، الگوریتم BFS:

- از رأس شروع s آغاز می‌کند
- ابتدا تمام رأس‌های با فاصله 1
- سپس فاصله 2
- سپس فاصله 3
- و به همین ترتیب پیش می‌رود

بنابراین اولین باری که به رأس t برسیم، کوتاه‌ترین فاصله را پیدا کرده‌ایم.

برای پیاده‌سازی:

- از **لیست مجاورت** برای نگهداری گراف استفاده می‌کنیم (به دلیل محدودیت‌های بزرگ n و m)
- یک آرایه distance نگه می‌داریم که فاصله هر رأس از s را ذخیره کند
- مقدار اولیه همه را -1 می‌گذاریم (به معنی بازدید نشده)
- فاصله s را برابر 0 قرار می‌دهیم
- از صف (Queue) برای اجرای BFS استفاده می‌کنیم

وقتی t را کشف کردیم، مقدار distance[t] پاسخ ماست.

اگر در پایان distance[t] همچنان -1 بود، یعنی مسیری وجود ندارد.

مراحل الگوریتم

1. خواندن n و m
2. ساخت لیست مجاورت
3. مقداردهی distance با -1
4. قرار دادن s در صف و distance[s] = 0
5. اجرای BFS:
    - تا وقتی صف خالی نشده:

- یک رأس u را خارج کن
- برای هر همسایه v:
- اگر بازدید نشده بود:

- distance[v] = distance[u] + 1
- اضافه کردن v به صف

6. در پایان:
   - اگر distance[t] == -1 → چاپ "NO POWER"
   - در غیر این صورت → چاپ distance[t]

پاسخ سوال چهارم:

فایل A4.py را اجرا کنید

طبق مطالب جزوه، برای تخمین شباهت Jaccard می‌توان به جای استفاده از مجموعهٔ کامل عناصر، از **امضای MinHash** استفاده کرد.

در این روش چندین تابع MinHash مستقل ایجاد می‌کنیم. در ساده‌ترین حالت، هر تابع MinHash با یک **جایگشت از عناصر مجموعه** ساخته می‌شود.

برای یک مجموعه A داریم:

h(A) = min index of element in A under permutation

به عبارت دیگر:

1. عناصر را طبق جایگشت مرتب می‌کنیم.
2. اولین عنصری که در مجموعه وجود دارد (مقدار 1 دارد) را پیدا می‌کنیم.
3. اندیس آن مقدار هش محسوب می‌شود.

اگر k تابع MinHash داشته باشیم، امضای هر مجموعه یک بردار k تایی است.

ویژگی مهم MinHash این است که:

احتمال برابر بودن هش دو مجموعه برابر با **شباهت Jaccard آن‌ها** است.

بنابراین اگر k تابع داشته باشیم:

approx_jaccard = (#hash_matches) / k

مراحل حل

1. خواندن ورودی‌ها (n، m، k).
2. ذخیره بردار دودویی مقاله‌ها.
3. خواندن k جایگشت.
4. برای هر مقاله:
    - برای هر جایگشت:
- بردار را بر اساس جایگشت مرتب کن
- اولین 1 را پیدا کن
- اندیس آن را در امضای MinHash ذخیره کن

5. برای مقالهٔ q:
    - شباهت تقریبی با همهٔ مقاله‌های دیگر را محاسبه کن.
6. مقاله‌ای که بیشترین شباهت را دارد انتخاب کن.
7. در صورت تساوی، کوچک‌ترین شماره مقاله انتخاب شود.

پاسخ سوال پنجم:

فایل A5.py را اجرا کنید

توضیح ایده حل

این مسئله یک **کوتاه‌ترین مسیر با تغییر وزن یال‌ها** است.

دایکسترا (طبق محتوای فایل 10_04) فقط با وزن‌های غیرمنفی کار می‌کند، بنابراین باید وزن مؤثر هر یال را قبل از اجرای دایکسترا در نظر بگیریم.

برای هر یال w:

اگر $w \le R$ → همان w

اگر $R < w$ → باید به قطعات R شکسته شود.

تعداد قطعات = $\text{ceil}(w / R)$

تعداد توقف = تعداد قطعات − 1

هزینه نهایی یال = w + (توقف‌ها × C)

دلیل این که هزینه پایه همان w است این است که مجموع پروازها همان w بوده، فقط بین آن‌ها توقف و پاکسازی اضافه می‌شود.

بعد از تبدیل هر یال به **یک وزن مؤثر مثبت**، دایکسترا استاندارد با مرتبه $O(V^2)$ قابل اجراست (طبق فایل آموزشی).

مراحل حل

1. ورودی را بخوانید.
2. برای هر یال، هزینهٔ مؤثر را حساب کنید.
3. گراف را با ماتریس یا لیست وزن‌ها ذخیره کنید.
4. الگوریتم دایکسترا را مطابق نسخهٔ آموزشی فصل «کوتاه‌ترین مسیر» اجرا کنید:
   - ابتدا تابع initialize_single_source را اجرا کنید تا مقدار اولیهٔ فاصله‌ها تنظیم شود.
   - در هر مرحله، رأسی از بین رأس‌های خارج از مجموعهٔ S انتخاب کنید که کمترین مقدار d را دارد.

- سپس تمام یال‌های خروجی آن رأس را با عمل relax آرام‌سازی کنید.

5. اگر مقدار فاصلهٔ مقصد برابر inf بود، یعنی مسیری وجود ندارد و باید -1 چاپ شود.
6. در غیر این صورت مقدار d[T] (کمترین هزینه رسیدن به مقصد) چاپ می‌شود.