MATH1005 SUMMARY

YUCHEN YAN U7663065

May 31, 2023

1 Sets

1.1 Power

The power set of A, denoted $\mathcal{P}(A)$ is the set of all subsets of A including $\emptyset$.

If A has n elements, $\mathcal{A}\ has\ 2^n$ elements.

1.2 Partition

$\mathcal{A} = \{\{1\}, P, C\}\ \{1\}, P, C\ are\ substes\ of \mathcal{A}$

Let S be a set and $\mathcal{A} \subseteq \mathcal{P}(S)$

$\mathcal{A}$ is a set, the elements of which are subsets of S. $\mathcal{A}$ is a partition of $\mathcal{S}$ if:

$\emptyset \notin \mathcal{A}$ $\forall s \in S \exists A \in \mathcal{A} s \in A$ the sets in $\mathcal{A}$ are pairwise disjoint.

2 A3: Relations and functions

2.1 Surjective

$$\forall b \in B\ \exists\ a \in A\ f(a) = b$$

Codomain and range are equal

2.2 Inverse function

$$aRb \Leftrightarrow bR^{-1}a$$

If and only if f is a **bijection** function from B to A

2.3 identity function

If $f : A \to B\ is\ a\ bijection, then f^{-1} \circ f = i_A and\ f \circ f^{-1} = i_B$

$f^{-1} \circ f = A \to B \to A = i_A$

$f \circ f^{-1} = B \to A \to B = i_B$

3 Digital Information

A digit is called a **bit**

A block of 8 bits called a **byte**

A block of 4 bits is called a **nibble**

3.1 Negative Integers

$(1d_1d_2d_3 \dots d_t)$ toggle all bits, add one, then negate.

3.2 Subtract

$x - y$ find the representation of $-y$, then use addition $x + (-y)$

3.3 Hexdecimal Multiplication

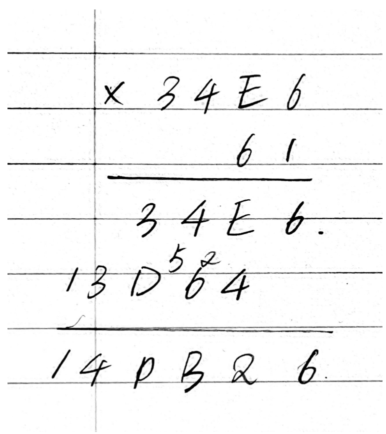

Figure 1: Hexdecimal Multiplication

4

B3: Matrices

4.1 Definition: Matrix

Let S be a set, and $m, n \in \mathbb{N}$

An $m \times n$ *matrix* (over S) is a rectangular array of members of S, the array having m rows and n columns.

The set of all $m \times n$ *matrix* matrices over S is denoted by $M_{m\times n}(S)$

5 Weighted Graphs

A weighted graph is a graph G together with a weight function weight : $E(G) \to \mathbb{Q}_+$.

Four type of problems in the weighted graphs:

1. Minimal spanning tree: Find a spanning tree of least possible total weight.
2. Find a Hamilton circuit of the least possible total weight.
3. Find a path between two given vertices that has the least possible total weight.
4. Maximum Flow

5.1 Kruskal's algorithm for minimal spanning tree

Input: Weighted connected graph G with n vertices.

Output: Minimal spanning tree T for G. Total weight W of this tree.

Method: Always select the minimum edge as long as there is no circuit formed when the new edge is added.

NOTE:This is an example of greedy algorithm and **always succeeds**.

5.2 The 'Nearest Neighbor' algorithm

Input: Weighted **complete** graph G with n vertices.

Output: Hamilton circuit for G as a list L of vertices. Total weight W of this circuit.

Method: Start with any vertex and choose a vertex v such that weight of the edge as small as possible. Repeat this step until all the vertices are included.

NOTE:This is an greedy algorithm and does not **always succeed**.

Example:

Question: There are 6 vertices and the graph is complete, how many possible Hamilton circuits exist?

With 6 vertices there are apparently $6! = 720$ circuits, but allowing for different starting points and directions of travel, only $5!/2 = 60$ are genuinely different.

5.3 Dijkstra's Algorithm

Input:

- Connected simple graph G. Vertices A,Z from G.
- Distance function dist:$E(G) \to \mathbb{Q}^+$

Output:

- Tree T containing A and Z as vertices.

- T is a subgrah of G. The unique path $A \to Z$ in T has minimal total distance of all paths $A \to Z$ in G.

- Labelling $L : V(T) \to \mathbb{Q}_+; L(v) = min.dist(A \to v)$.

Method: Start with the vertex A and find the adjacent vertices. If the vertex is unmarked, then mark them. If the vertex is marked select one with minimum value as the next current vertex.
NOTE: Vertex is locked in implies that we have found the path of minimum distance from the starting vertex A. The **fringe vertices** are those have been marked but not yet locked in. The spanning tree produced by Dijkstra's algorithm will not be minimal in general.

5.4 Transport Networks

Flow: The flow F is a function $E(D) \to \mathbb{Q}_+$, where D is the digraph representing the network. However, each edge e has a fixed capacity $C(e)$ whereas its flow $F(e)$ can vary, subject to constraints:

1. Flow cannot exceed capacity. $[\forall e \in E(D) F(e) \leq C(e).]$

2. In each edge, flow direction = edge direction

3. Total flow into a node equals total flow out, except for nodes s't (source, terminal).
   $[\forall v \in V(D) \backslash \{s, t\} \sum_{e \in v_{out}} F(e) = \sum_{e \in v_{in}} F(e)]$

Input: Transport network D with capacity function C. **Output:** A Maximum flow function F_{MAX} for the network.
Method:
NOTE: Always go through the minimum terminal labelling first.

5.4.1 Applications of Transport Network in Matching Problems

To the arrow diagram for the relation add an extra node s, the 'super source', and a link from s to each member of the domain of the relation. Then add another extra node t, the 'super sink' (super target), and a link to t from every member of the codomain of the relation. Now give all links a capacity of 1. A maximum flow now gives a maximal matching by picking all arrows from domain to codomain that have non-zero flow. Remarks (not necessary for the answer to this question but relevant to the presentation of the solutions to the remaining questions):

1. Since all links have capacity 1, it is not necessary to display capacities.

2. Since a link with capacity 1 is either fully used or not used at all, rather than show spare capacity and flow we can show used/unused.

3. Similarly, there is no need to show potential flow values on vertex labels.

4. My convention will be show used links by heavy lines.

5. When a link AZ from domain to codomain is used, the links sA and Zt must also be used, so there is no logical need to mark them. However, marking them helps to avoid making mistakes in annotating vertices, so I will still mark them, but less thickly.

NOTE: Start with codomain in alphabetical order.

6 D3: Random Walks on Graphs

7 Definition:Random Walks

Let G be a digraph with n vertices $V = V(G) = \{1, \ldots, n\}$ and directed edge set $E = E(G)$.

A square matrix with non-negative entries such that the enties in each row sum to 1 is called a stochastic matrix.

Basis vectors : elementary vectors. For any given n, let B_n denote the set of basis vectors $\{e_1, \ldots, e_n\}$ where e_i is the $n \times 1$ vector with 1 as the i-th entry and all other entries zero.

For $X_0 = e_i \in B_n$ the sequence $(X_k)_{k \in \mathbb{N}^*}$ specified by G and T is called the random walk on G starting at vertex i with transition matrix T.

Then $X_k = (T')^k e_i = (q_j)_{1 \leq j \leq n}$ say gives, for $1 \leq j \leq n$, the probability q_j of being at the vertex j after k steps, starting form vertex i.

8 Method of Calculating Steady States

1. Write out the transition matrix T.

2. Transpose the matrix T to T'.

3. Subtract the identity matrix from the transposed matrix T'.

4. Add columns with all entries 0 after the last column of matrix.

5. **Replace** the last tow with all 1's

6. Apply the row operations to reduce the matrix to row reduced echelon form and the last column is the steady states.

8.1 HYPOTHESES OF THE MODEL

1. Each link to W is saying that W is a bit awesome.

2. A link to W from an awesome page is saying more than a link to W from a less awesome page.

8.2 The Structure of an Internet

Represent an internet as a digraph called **webgraph**. The vertices represent of the web, and there is a directed edge from verttex X to vertex Y if and only if there is a hyperlink from the page corresponding to X to the page corresponding to Y.

8.3 Introduce Damping Factor

1. With the probability α, the RS will type in the URL of a randomly chosen page(unforced teleporting).

2. With probability $(1-\alpha)$ the RS will proceed as follows: if there is at least one hyperlink on the current page, the RS will choose at random one of the these hyperlinks and click on it; if there is no hypoerlink on the current page, then the RS will choose a new page at random(forced teleporting).

8.4 PageRank

Let $P = (p_1, p_2, \ldots, p_n)$ be the steady state vector for the random walk on W using the transition matrix M. For each $i \in \{1, \ldots, p_i\}$, p_i is the PageRank of page i.

8.5 Modified Probability for Transition from vertex i to j

$$\begin{aligned} m_{ij} &= \alpha/n + (1-\alpha)t_{ij} && (1) \\ M = (m_{ij})_{1\le i,j\le n} &= (\alpha/n + (1-\alpha)t_{ij})_{1\le i,j\le n} && (2) \\ &= (\alpha/n)U + (1-\alpha)T && (3) \end{aligned}$$

8.6 Calculating PR

$$(I - (1-\alpha)T')PR = (\alpha/n)1 \tag{4}$$

8.7 iterative Approximation Method

$$P_0 = (1/n)1; P_k = \alpha P_0 + (1-\alpha)T'P_{k-1}, k \ge 1. \tag{5}$$

9 C3:Markov Process

It is closely related with the above topic.

10 Steady State vector

$$T'v = v \tag{6}$$

Theorem 1 *Let* $T = (T_{ij})_{1 \le i,j \le k}$ *be the transition matrix for a* k *state Markov process with state vectors* $x_n, n \in \mathbb{N}$*. Then* $\forall n \le 1$*:*

1. $x_n = T'x_{n-1}$
2. $(T^n)_{ij} is\ the\ n - step\ i - to - j\ transition\ probability.$
3. $x_n = (T')^n x_0$