Playing Games with my Heart

The following games are all in terms of dollars and money. If it makes you feel better, replace all occurrences of "dollars" with "love" (or imagine that you can exchange one for the other).

Problem 1: You're playing the following game: on a table is a pile of money, initially containing a single dollar. On every turn, you have two options. The first option is to take the money on the table, ending the game. The second option is to roll a die. If you roll a 6, all the money on the table disappears and the game ends. Otherwise, a dollar gets added to the pile of money on the table and the game continues. What's the optimal strategy and EV for this game?

Problem 2: You're playing a game where you try to throw a coin of radius R onto a table lined with a grid where each cell is $D \times D$. The goal is to get as close to the middle of a grid cell as you can. Specifically, your payout in dollars is the distance from the coin to the grid – if the coin overlaps the grid, you get \$0. Assuming you aren't very good at this game, and you can guarantee your coin lands on the table and grid, but have no control over where in a cell it lands, what's your expected value from this game in terms of R and D?

Problem 3: You've discovered a very lucrative gambling opportunity! Basically, you can bet as much money as you want on a weighted coin that flips heads with probability $p > 0.5$. If you win the flip (i.e., call it correctly), your bet is doubled. If you lose the flip, you lose your bet. You will play this game once every day for the next 10 years. How should you choose your bets in order to maximize your long-term profit? *Hint:* Think about your expectation over many plays of the game with some betting strategy, and try to maximize that.

Algorithms for Finding Love

Problem 4: (Online Bipartite Matching Problem) You have created an app called GTinder. You are given a bipartite graph $G = (L, R, E)$ where L is the set of vertices on the "left" side, R is the set of vertices on the "right" side, and E are edges connecting a vertex from L to a vertex in R. You want to find the maximum matching M (the maximum subset of edges so that no two edges share a vertex) in G so that your customers are satisfied. However, vertices in R (and the neighbors of that vertex) are only revealed one-by-one, so you must immediately decide to match that vertex or not. Show that the Greedy algorithm (make a match if an edge can be added to our matching M) yields a 2-approximation (Greedy takes at least half as many matches as the maximum matching for any input bipartite graph G).

Remark: No deterministic algorithm can do better than this 2-approximation!

Problem 5: Suppose you have a very large file containing the names of potential Valentine's Day dates. You want to choose one at random. However, since the file is so large (you're very popular), you can't fit it all into memory, and you don't know how exactly how long the file is. You're only allowed to read the file one name at a time, and only once. Design an algorithm to randomly choose a name under these constraints.

Followup: What if you want to randomly choose k names?

Quantum Algorithms

We describe some elementary quantum computing here for problem 5. A list of definitions for reference:

1. We define a *ket* $|x\rangle$ where $x \in \mathbb{N}$ is an n-bit value. $|x\rangle$ corresponds to a vector $e_x \in \mathbb{C}^n$ that has a 1 in the x index and 0 everywhere else. For example, $|0\rangle = (1, 0)$ and $|1\rangle = (0, 1)$. Another example: $|000\rangle = (1, 0, 0, 0, 0, 0, 0, 0)$ and $|101\rangle = (0, 0, 0, 0, 0, 1, 0, 0)$. Clearly, the vector representation is not useful.
2. A linear combination of kets defines a state $|\psi\rangle$. For example, a state would be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. We define the coefficient of a ket in a state as a *probability amplitude*. If a state has n kets with probability amplitudes $p_1, ..., p_n \in \mathbb{C}$ we require that $|p_1|^2 + ... + |p_n|^2 = 1$ (Born's rule). For example $\frac{i}{\sqrt{2}}(|0\rangle + |1\rangle)$ is a valid state but $\frac{1}{\sqrt{3}}(|0\rangle + |1\rangle)$ is not.
3. We define a ket as a *pure state*. A linear combination of multiple kets is a *mixed state*.
4. The measurement of a mixed state collapses the state to a pure state. The probability of a mixed state collapsing to a specific pure state with probability amplitude p is $|p|^2$.
5. If we have two kets $\alpha|x\rangle$ and $\beta|y\rangle$, we define their Kronecker product $\alpha|x\rangle \otimes \beta|y\rangle = \alpha\beta|xy\rangle$. As a shorthand, this is written as $(\alpha|x\rangle)(\beta|y\rangle)$.
6. A qubit is represented as a linear combination of $|0\rangle$ and $|1\rangle$. n qubits are represented as the Kronecker product of those state.
7. A quantum gate that operates on n qubits is represented by a unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$; a unitary matrix has the property that $U^{-1} = U^*$ where U^* is the conjugate transpose of U.

We define elementary quantum gates below:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{Hadamard gate} \tag{1}$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{X (or NOT) gate} \tag{2}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{Z gate} \tag{3}$$

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{CX (or CNOT) gate} \tag{4}$$

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad \text{CZ gate} \tag{5}$$

To simplify further, we show the outcome of each gate on individual kets:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{6}$$
$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{7}$$
$$X|0\rangle = |1\rangle \tag{8}$$
$$X|1\rangle = |0\rangle \tag{9}$$
$$Z|0\rangle = |0\rangle \tag{10}$$
$$Z|1\rangle = -|1\rangle \tag{11}$$
$$CX|0s\rangle = |0s\rangle \tag{12}$$
$$CX|1s\rangle = |1\rangle X|s\rangle \tag{13}$$
$$CZ|0s\rangle = |0s\rangle \tag{14}$$
$$CZ|1s\rangle = |1\rangle Z|s\rangle \tag{15}$$

Also, some identities are that $(HXH)|\psi\rangle = Z|\psi\rangle$ and $(HCXH)|\psi\rangle = CZ|\psi\rangle$ (and vice versa). Finally, we note that all of these gates are self-inverses.

For future reference: https://en.wikipedia.org/wiki/Quantum_logic_gate.

Problem 5: We define the Bernstein-Vazirani problem as follows. Given a function $f(x) = s \cdot x$ where $s, x \in \{0,1\}^n$ and $s \cdot x$ is the dot product between strings where $s \cdot x = s_1x_1 + s_2x_2 + ... + s_nx_n \pmod 2$, we want to find s (the secret string) by querying f.

Part A: Give a classical algorithm for this problem that runs in $O(n)$.

Part B: A quantum algorithm for Bernstein-Vazarani only requires 1 query to f. We show an example below for $n = 4$ with $s = 1101$ (on the board). The algorithm has the following steps:

1. Initialize all qubits to $|0\rangle$ (state is $|0^n\rangle$).
2. Execute H on all qubits.
3. Execute U_f where $U_f|x\rangle = (-1)^{s \cdot x}|x\rangle$.
4. Execute H on all the qubits again.
5. Measure all the qubits.

What is the output for the $n = 4$ BV circuit?

Part C: (challenge) Prove that the BV algorithm always outputs s, the secret string.

Part D: U_f is actually implemented as a sequence of CX gates with an ancilla qubit (shown on the board). Rewrite the $n = 4$ BV circuit using no CX gates (hint: use CZ gates).

Part E: Let $BV(n)$ be a BV circuit with $n + 1$ qubits and a secret string $s = 2^n - 1$.

On quantum computers, qubits in a circuit are mapped onto hardware qubits. We call the circuit qubits as *virtual qubits* and the hardware qubits as *physical qubits*. In order to execute 2-qubit gates in hardware, virtual qubits must be on adjacent physical qubits. We can rearrange virtual

qubits by performing SWAP gates (example shown on the board).

This is called *qubit routing*, which is an NP-Complete problem in general. For $BV(n)$, we know the minimum number of SWAPs required for execution. Compute the minimum number of SWAPs for $BV(5)$ on a linear backend. Suppose you can rearrange the virtual qubits manually before program execution. What is the minimum number of required SWAPs now? Is there a general formula for $BV(n)$?

Randomized 2SAT

Problem 6: We describe a randomized 2SAT algorithm:

1. Initialize $x_1, ..., x_n$ as a random assignment.
2. While $\phi(x_1, ..., x_n) \neq 1$, choose a unsatisfied clause $(x_i \vee x_j) = 0$. Flip x_i or x_j (but not both).
3. Return $x_1, ..., x_n$.

We ask the following. Let X_i be the number of matching assignments between $x_1, ..., x_n$ and some assignment $x_1^*, ..., x_n^*$ that satisfies ϕ (we do not know x_i^*).

Part A: Suppose we execute line (2). What is

$$\mathbb{P}(X_{i+1} = j + 1 | X_i = j) \tag{16}$$

and

$$\mathbb{P}(X_{i+1} = j - 1 | X_i = j) \tag{17}$$

for $j \geqslant 1$. What is $\mathbb{P}(X_{i+1} = 1 | X_j = 0)$?

Part B: Suppose we tighten the probabilities found in Part A to get a Markov chain. Let $h(j)$ be the expected number of variable flips (in line 2 of the algorithm) until we terminate (on line 3). Compute a closed form expression for $h(j)$. Hint: define each $h(j)$ as a recursion and solve a system of linear equations.

Part C: What is the running time of this algorithm if we want to succeed with probability $1 - \epsilon$?