Quantathon 2025 - Team >:3

Arjun Patel, Max Gao, Vincent Qi, Harsh Akunuri

February 2025

1 Problem 1

1.1 Problem 1a

We claim that $\mathbb{E}[W_N] = 0$. Using the linearity of expectations we can break the total expected winnings down into the expected winnings of each round and then sum them back up. This is a technique we'll use throughout the problem set.

In round 1, you bet 1 dollar and there is probability 1 of U_1 being rank 1, so your expected total value is $1 - 1 = 0$.

In round 2, U_2 is 1 of 2 numbers chosen uniformly from the same distribution. Hence, there is a $\frac{1}{2}$ chance of it being rank 1 and a $\frac{1}{2}$ chance of being rank 2, so your expected earnings are $\frac{1}{2} \cdot 1 - \frac{1}{2} = \frac{1}{2}$. Since you pay $\frac{1}{2}$ to play, in round 2 you can expect to earn 0.

In round 3, U_3 is 1 of 3 numbers chosen from this $[0, 1]$ distribution so there is a $\frac{1}{3}$ chance it is the maximum value and $\frac{2}{3}$ that it is not. So again, your expected earnings are $\frac{1}{3} \cdot 1 + \frac{2}{3} \cdot 0$ and the pay is $\frac{1}{3}$. So, your earnings are $\frac{1}{3} - \frac{1}{3} = 0$.

Finally, in round 4, U_4 is 1 of 4 numbers chosen from among this same distribution so there is a $\frac{1}{4}$ chance it is the max and $\frac{3}{4}$ that it is not. So, your net gains are $\frac{1}{4} \cdot 1 + \frac{3}{4} \cdot 0 - \frac{1}{4} = 0$.

So, $E[W_4] = 0 + 0 + 0 + 0 = 0$

Observe in general that on the nth round, U_n has a $\frac{1}{n}$ chance of being rank 1 and a $\frac{n-1}{n}$ chance of not. Since you pay $\frac{1}{n}$ to play, your expected value is $\frac{1}{n} \cdot 1 + 0 \cdot \frac{n-1}{n} - \frac{1}{n} = \frac{1}{n} - \frac{1}{n} = 0$ net earnings from that round. This means that our result holds for any positive natural N. For example, $\mathbb{E}[W_{10}] = 0$ under this strategy.

1.2 Problem 1b

We claim there is not a way to beat the above strategy and that your expected value under any strategy will be 0. One simple intuition for this is in round n, your expected winnings given that you play are 0, as mentioned in $1a$. However, if you pass, you are guaranteed to win 0. Summing up the expected values of each round, in all cases your expected value by the end of the game will be 0.

1.3 Problem 1c

We can extend $1a$ and $1b$ to N arbitrarily large, like $N = 2025$. As seen in $1b$, 0 is an upper bound on the expected value of all possible strategies. As seen in $1a$, playing every round leads to winnings an expected value of 0, so that upper bound is achieved with this strategy. So the largest possible value is $\mathbb{E}[W_N] = 0$.

2 Problem 2

2.1 Problem 2a

We present a general strategy to solve $2a$ and $2b$.

For each $n = 0, 1, ..., N-1$, define $m_0 = 0$ and define $m_n = \max(U_1, U_2, ..., U_n)$ for $n > 0$.

On round $n+1$, our expected value for winning from playing in that round is as follows:

- Suppose $U_{n+1} > m_n$. Then we will receive \$1. This occurs with probability $1 - m_n$.
- On the other hand, it may be that $U_{n+1} < m_n$. Then we'll earn 0 dollars. This occurs with probability m_i.
- On round $n+1$, we always pay $\frac{-1}{n+1}$ to play the game.
- Therefore, our total expected earnings are

$$(1 - m_n)(1) + 0(m_n) - \frac{1}{n+1}$$

$$= 1 - m_n + \frac{-1}{n+1}$$

Our only two options are to play or pass. We can maximize our expected value if in every round, we choose the option that gives us the higher expected value because our decision in each round is independent. Notice that the expected value of winnings from passing is just 0. So, we decide to play on round $n+1$ if $1 - m_n + \frac{1}{n+1} > 0$.

That is, our strategy is as follows:

- On round $n+1$, play if $\frac{n}{n+1} > m_n$.
- Else, pass.
- Do this for $n = 0, 1, ..., N-1$.

If we do this for every round until the end of the game, we'll have maximized our expected value.

In the $N = 4$ case for problem 1, we can say our strategy specifically is

- On round 1, always pass, as $\frac{0}{1} > 0$ is not true. Note we expect 0 earnings from round 1 from either passing or playing.
- On round 2, play if $\frac{1}{2} > m_1 = U_1$. Else pass.
- On round 3, play if $\frac{2}{3} > m_2 = \max(U_1, U_2)$. Else pass.
- On round 4, play if $\frac{3}{4} > m_3 = \max(U_1, U_2, U_3)$. Else pass.

We claim that this strategy gives higher expected winnings than $1a$. That is, we claim that our strategy has a positive expected value.

Notice that in each round, our expected value of earnings from that round is always nonnegative, since we always are picking the option that gives us an expected value of $\max(0, 1 - m_n - \frac{1}{n+1})$. So, the overall expected value of this strategy is non-negative. Thus, if we can show that the expected value of this strategy of a specific round is strictly positive, we know that the overall expected value of this strategy is strictly positive.

Notice in round 2 we decide to play if $\frac{1}{2} > U_1$.

Case 1: $U_1 < \frac{1}{2}$. This occurs with chance $\frac{1}{2}$. Then we decide to play. We can expect to earn $1 - U_1 - \frac{1}{2} > 0$, since $U_1 < \frac{1}{2}$.

Case 2: $U_1 > \frac{1}{2}$. This occurs with chance $\frac{1}{2}$. Then we pass and earn 0 for sure.

$U_1 < \frac{1}{2}$ or $U_1 > \frac{1}{2}$ is an exhaustive casing, so our total expected value for round 2 is $\frac{1}{2}(1 - U_1 - \frac{1}{2}) + \frac{1}{2} \cdot 0 = \frac{1}{2}(1 - U_1 - \frac{1}{2}) > 0$. Hence, we have a strictly positive expected value on round 2.

Since every other round contributes a non-negative amount of earnings, we conclude this strategy has a strictly positive expected value, as desired.

2.2 Problem 2b

We use the same strategy as seen in $2a$. As mentioned in our original exposition of the strategy, since we are maximizing the expected value of the earnings from each round, this strategy should achieve the largest possible expected winnings for the overall game.

We claim when $N = 2025$ the expected value of this strategy is roughly 2.507. In exact notation this is

$$\sum_{i=1}^{2024} [(\frac{i}{i+1} - (\frac{i}{i+1})^2) \cdot (\frac{i}{i+1})^i]$$

We'll break this formula down below:

In order to get the total expected value of winnings from the game, we sum the expected value of winnings from each round $1, 2, ..., 2025$ under our strategy. We'll ignore round 1 since we know we can expect to earn 0 in that round.

In each strategy, we either play and earn some positive expected amount or pass and gain 0.

On round $i + 1$, we look at the previous i rounds to determine whether we pass or play. We only choose to play if every number picked so far has been below $\frac{i}{i+1}$. This occurs with probability of picking a number i times from $[0, 1]$ such that the number is less than $\frac{i}{i+1}$ every time. This probability is $[\frac{i}{i+1}]^i$. Then, the expected max value among these i numbers in $[0, \frac{i}{i+1}]$ should be $(\frac{i}{i+1})^2$ since it is a known fact that the expected value the max of uniformly choosing n values over an interval $[0, k]$ is $\frac{nk}{n+1}$. Recall from $2a$ that the expected earnings of playing is $1 - m_i - \frac{1}{i+1}$, where m_i is the max of $U_1, U_2, ...U_i$. We

can substitute in our expected max value to see that the payout is $\frac{i}{i+1} - (\frac{i}{i+1})^2$. When we don't play, we are passing and have an expected value of 0.

Therefore on round $i+1$ our expected value for winnings from the round is

$$[\frac{i}{i+1} - (\frac{i}{i+1})^2] \cdot (\frac{i}{i+1})^i + 0,$$

and this is the term we see in our summation.

Additionally, we tried determining the behavior of our expected value function as N, the number of rounds, approaches infinity, and found that the following function produces Figure 1:

$$y = \sum_{n=1}^{x} \left(\left(\frac{n}{n+1} \right)^{(n+1)} - \left(\frac{n}{n+1} \right)^{(n+2)} \right)$$

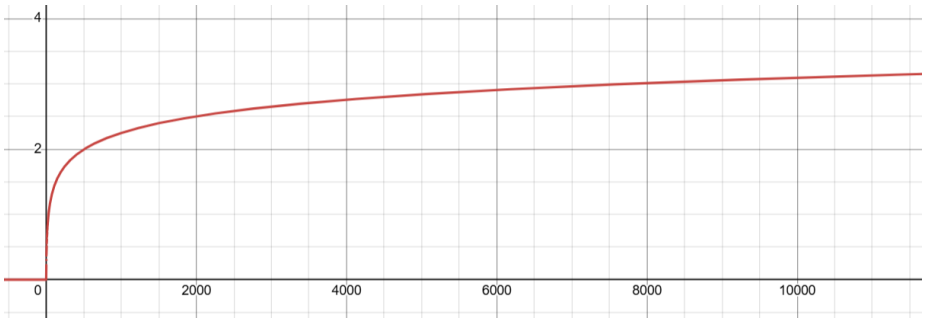

Figure 1: The Game as N Tends to Infinity

We thus find that the value grows logarithmically with respect to the number of rounds.

3 Problem 3

3.1 Problem 3a

In this variant, unlike the first two games, it is possible for the expected payoff of playing in round n to be negative, with the long-term expected continuation value for playing being positive. To capture that trade-off, we treat the game as an optimal-stopping problem.

We first define our value function as

$$V_n(m_{n-1}) = \begin{cases} \max(0, \mathbb{E}[\text{payoff}_n + V_{n+1}(\max(m_n)]) & \text{if } n \in \{1, 2, \ldots, n-1\}, \\ \max(0, \mathbb{E}[\text{payoff}_n]), & \text{if } n = N, \end{cases}$$

where payoff_n is the random variable denoting the net gain (positive or negative) obtained by playing in round n. We see that the recursive definition captures the potential gains or losses from continuing to play. Note that V_n does not have a continuation term as it is the last round of play.

We know that

$$\text{payoff}_n = \begin{cases} 1 - \frac{1}{n}, & \text{if } U_n > m_{n-1}, \\ -\frac{1}{n}, & \text{if } U_n < m_{n-1}, \end{cases}$$

so the expectation for playing is $\mathbb{P}(U_n > m_{n-1})(1 - \frac{1}{n}) + \mathbb{P}(U_n < m_{n-1})(-\frac{1}{n})$.

Since U_n is uniformly chosen in $(0, 1)$, $\mathbb{P}(U_n > m_{n-1}) = 1 - m_{n-1}$ and $\mathbb{P}(U_n < m_{n-1}) = m_{n-1}$. Hence,

$$\mathbb{E}[\text{payoff}_n] = (1 - m_{n-1})(1 - \frac{1}{n}) - \frac{m_{n-1}}{n} = \frac{n-1}{n} - m,$$

so $\mathbb{E}[\text{payoff}_n + V_{n+1}(\max(m_n)]) = \frac{n-1}{n} - m + \mathbb{E}[V_{n+1}(m_n)])$ by linearity of expectations.

Furthermore, the expected continuation value is $\mathbb{E}[V_{n+1}(m_n)]) = \int_0^1 V_{n+1}(m_n)du$, obtained by averaging u from $\text{Unif}(0, 1)$.

Putting this all together gives

$$V_n(m_{n-1}) = \begin{cases} \max(0, \frac{n-1}{n} - m + \int_0^1 V_{n+1}(m_n)du) & \text{if } n \in \{1, 2, \ldots, n-1\}, \\ \max(0, \frac{n-1}{n} - m), & \text{if } n = N. \end{cases}$$

Hence, we can directly find the expected winnings by computing $V_1(0)$ with $N = 3$:

$$
\begin{aligned}
V_1(0) &= \max(0, \frac{0}{1} - 0 + \int_0^1 V_2(u)du) \\
&= \int_0^1 V_2(u)du \\
&= \int_0^1 [\max(0, \frac{1}{2} - u + \int_0^1 V_3(\max(u,v))dv]du \\
&= \int_0^1 [\max(0, \frac{1}{2} - u + \int_0^u V_3(u)dv + \int_u^{\frac{2}{3}} V_3(v)dv + \int_{\frac{2}{3}}^1 V_3(v)dv)]du \\
&\qquad\qquad \text{(Split integral at break points)} \\
&= \int_0^1 [\max(0, \frac{1}{2} - u + u(\frac{2}{3} - u) + \int_u^{\frac{2}{3}} (\frac{2}{3} - v)dv + \int_{\frac{2}{3}}^1 0dv)]du \\
&= \int_0^1 [\max(0, \frac{1}{2} - u + \frac{2u}{3} - u^2 + (\frac{4}{9} - \frac{2}{9}) - (\frac{2u}{3} - \frac{u^2}{2}))]du \\
&= \int_0^1 [\max(0, -\frac{u^2}{2} - u + \frac{13}{18})]du \\
&= \int_0^{\frac{\sqrt{22}-3}{3}} [-\frac{u^2}{2} - u + \frac{13}{18}]du \qquad (u = \tfrac{\sqrt{22}-3}{3} \implies -\tfrac{u^2}{2} - u + \tfrac{13}{18} < 0) \\
&= -\frac{(\frac{\sqrt{22}-3}{3})^3}{6} - \frac{(\frac{\sqrt{22}-3}{3})^2}{2} + \frac{13 \cdot \frac{\sqrt{22}-3}{3}}{18} \\
&\approx 0.2183845
\end{aligned}
$$

We conclude that the maximum total expected payoff for $N = 3$ is 0.2183845.

3.2 Problem 3b

For $N = 2025$ rounds, solving the nested integral recursion would require on the order of 2^{2025} operations, which is both infeasible by hand and using computer technology. Instead, we implement a Monte Carlo backward-induction solver to evaluate each integral $\int_0^1 V_{n+1}(m_n)du$ by approximating the shape of every value function $V_n(m)$.

To make our backward-induction implementation more convenient, we switch to the following, algebraically equivalent, form of the value function:

$$
V_n(m) = \max\Big\{ 0,\ \frac{n-1}{n} - m\ +\ \int_0^1 V_{n+1}\big(\max(m,u)\big)\, du \Big\}, \quad V_{2026}(\cdot) \equiv 0.
$$

Next, we discretize $(0, 1)$ by drawing S independent samples $u_1, \ldots, u_S \sim \text{Unif}(0, 1)$. Rather than explicitly solving for $\int_0^1 V_{n+1}(\max(m, u))du$, we approximate the integral by computing $\frac{1}{S}\sum_{i=1}^S V_{n+1}(\max(m, u_i))$. We then perform backward

induction, starting from $n = N$ to $n = 1$, recursively constructing $m \mapsto V_n(m)$ at every step using payoff_n and $V_{n+1}(m)$. Finally, our estimate $V_1(0)$ gives us our largest expected payoff.

The code used for this simulation is attached in the appendix. Below is a line-by-line breakdown:

- Lines 9-10: Defining parameters for number of rounds and samples.
- Lines 12-14: Setting up our random number generator.
- Lines 16-18: Generating and sorting `u`, our vector of S randomly generated points over $\text{Unif}(0, 1)$.
- Lines 20-22: Initializing variables `V_cur`, `V_new`, `prefix`, and `EV_start`. At round n, `V_cur`$[i] = V_{n+1}($`u`$[i])$, `V_new`$[i] = V_n($`u`$[i])$, and `prefix` will be a prefix array for `V_cur`. We zero-initialize `V_cur` since $V_{N+1}(\cdot) = 0$.
- Lines 24-40: Backward-induction from $n = N$ to 1. In each iteration, we build `V_new` from `V_cur`.
  - Lines 25-26: Build the prefix-sum array `prefix`$[k] = \sum_{i=0}^{k-1} V_{n+1}($`u`$[i])$.
  - Lines 29-35: Traverse `u` to build `V_new`. For each $i \in \{0, \ldots, S-1\}$:
    - Line 30: Compute the immediate payoff `ev_now` $= \frac{n-1}{n} -$ `u`$[i]$.
    - Line 31: Compute the expected continuation value `cont` $= \frac{(i+1)\texttt{V\_cur}[i] + (\texttt{prefix}[S] - \texttt{prefix}[i+1])}{S}$.
    - Lines 32-33: Set `V_new`$[i] = \max(0,$ `ev_now` $+$ `cont`$)$.
  - Line 38: When $n = 1$, set `EV_start` $= \frac{1}{S}\sum_{i=0}^{S-1} V_2($`u`$[i]) \approx \int_0^1 V_2(u)du$. Since $\text{payoff}_1 = 0$ and `EV_start` ≥ 0, it follows that `EV_start` $\approx \int_0^1 V_2(u)du = \max(0, \text{payoff}_1 + \int_0^1 V_{n+1}(m_n)du) = V_1(0)$.
  - Line 39: Swap `V_cur` and `V_new` to set `V_cur`$[i] = V_n($`u`$[i])$ for round $n - 1$.
- Line 42: Output `EV_start`.

Running the code with $S = 10^8$ and $N = 2025$ yields $V_1(0) \approx 1.49914$. Hence, we conclude that the largest possible expected winnings is about 1.49914. By the law of large numbers, this Monte Carlo estimate converges to the true value as $S \to \infty$.

3.3 Problem 3c

3.3.1 Optimal strategy

The optimal strategy is a threshold rule: at each round n you play only if the current maximum m_n lies below some critical point t_n. In our C++ algorithm for 3b, uncommenting lines 28, 34, and 37 generates the vector `threshold`, with

`threshold`$[n-1] = t_n$, where $m_{n-1} \in (0, t_n) \iff \mathbb{E}[\text{payoff}_n + V_{n+1}(n)] \geq 0$. For round n, `threshold`$[n-1]$ is set as the largest $i \in \{0, \ldots, S-1\}$ such that `cont` + `ev_now` ≥ 0. As $S \to \infty$, `threshold` becomes increasingly accurate by the law of large numbers.

3.3.2 Asymptotic analysis

Using the C++ code for part 3b, in addition to the main run at $N = 2025$ and $S = 10^8$, we also simulated various games for N from 1 and 2000 with $S = 10^7$ and for N from 2000 and 2500 with $S = 10^8$. These simulations help trace the curve of $N \mapsto V_1(0)$ shown in Figure 2. From the graph, we can see that $V_1(0)$ grows logarithmically compared to N. A logarithmic regression (the red line)

$$y = 0.185647 \log(x - 0.613536) + 0.130738$$

fits the data well, achieving an R^2 value of 0.9916. Hence, as $N \to \infty$, our largest expected winnings also goes to ∞.

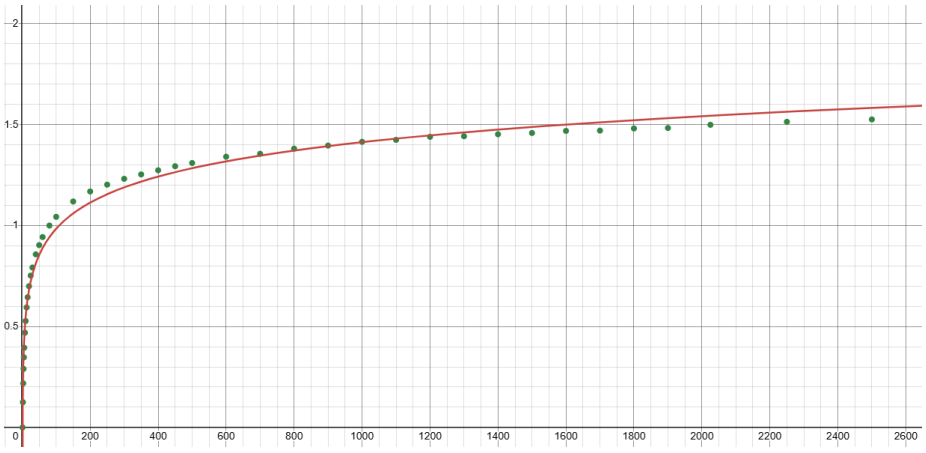

Figure 2: Simulated Games with Logarithmic Regression

3.3.3 Financial interpretation

One nice interpretation of this problem is we have a portfolio which at times $n = 1, 2, ...N$ will have value $U_1, U_2, ..., U_n$. "STOP" corresponds to selling of the portfolio and exiting with some accumulated earnings, and paying $\frac{1}{n}$ at each time step to continue to hold the portfolio until the next time step ("PLAY") corresponds to some kind of fee. If we assume the fees are fairly constant over time, it also makes sense that the time-0 value of the fees decrease as time increases because they are being discounted from farther in the future back to the present. We are of course interested in studying the behavior of this setup and determining how to maximize our expected earnings. However, it's also

important to understand the variation in the game, not just the expected value. This is corresponding to some level of volatility and risk in the portfolio. We can adapt our computer simulation to include these statistics.

Appendix: C++ Simulation for problem 3

```cpp
#include <random>
#include <algorithm>
#include <vector>
#include <numeric>
#include <iostream>
using namespace std;

int main(){
    const int N = 2025;
    const int S = 100000000;

    random_device rd;
    mt19937_64 gen(rd());
    uniform_real_distribution<double> dist(0.0, 1.0);

    vector<double> u(S);
    for(int i = 0; i < S; ++i) u[i] = dist(gen);
    sort(u.begin(), u.end());

    vector<double> V_cur(S, 0.0), V_new(S), prefix(S+1);
    vector<double> thresholds(N+1);
    double EV_start = 0.0;

    for(int n = N; n >= 1; --n){
        prefix[0] = 0.0;
        for(int i = 0; i < S; ++i) prefix[i+1] = prefix[i] +
             V_cur[i];

        // int thresh_idx = -1;
        for(int i = 0; i < S; ++i){
            double ev_now = double(n - 1) / n - u[i];
            double cont = ((i + 1) * V_cur[i] + (prefix[S] -
                 prefix[i + 1])) / S;
            double F = ev_now + cont;
            V_new[i] = (F > 0.0 ? F : 0.0);
            // if(F >= 0.0) thresh_idx = i;
        }

        // thresholds[n] = (thresh_idx >= 0 ? u[thresh_idx]
            : 0.0);
        if(n == 1) EV_start = prefix[S] / S;
        swap(V_cur, V_new);
    }

    cout << "Expected total gain: " << EV_start << "\n\n";
    return 0;
}
```