Jackson Eshbaugh

CS 202: Quiz 8 *(P vs. NP)*

1. What is the P vs. NP problem? What is the history behind this problem, and what is its current status?

The P vs. NP problem is a fundamental question about algorithmic efficiency in computer science. The problem asks if all problems that can be verified quickly—those which are class NP *"nondeterministic polynomial time"* problems—can be reduced to class P problems, those solvable in polynomial time. Despite much effort towards a solution, the problem remains open.
The problem's genesis came in 1956, when Kurt Gödel, an Austrian logician, penned a letter to John von Neumann, describing the P vs. NP problem [1]. It is not known if von Neumann ever read the letter, but it was later discovered and widely distributed in 1988. However, P vs. NP truly became a phenomenon when Richard Karp published his 1972 paper which proved that various combinatorial problems are NP-complete.
Currently, the P vs. NP problem remains unsolved. It is a **Clay Milenium Prize Problem**, which means that the Clay Mathematics Institute will reward a solution with $1 million [2].

2. What would the consequences be if $P = \mathrm{NP}$? Conversely, what if $P \neq \mathrm{NP}$?

If it were to be proven that $P = \mathrm{NP}$, then every problem that can be verified quickly could also be solved quickly. This would be revolutionary, changing many things in the field of computer science. For example, RSA encryption (which relies on the difficulty of factoring large numbers) could no longer be considered secure because it is an NP algorithm—verification of an RSA key is quick, and if $P = \mathrm{NP}$, then reversing the process would also be quick [3].
A proof that $P \neq \mathrm{NP}$ would mean that not every problem whose solutions can be efficiently verified could be efficiently solved. This would mean that some problems would just be intractable, perhaps RSA. Note this distinction: the logical statement that $P \neq \mathrm{NP} \Rightarrow$ the RSA agorithm is intractable in the reverse direction **is not necessarily true**. Since P and NP are sets of problems, some problems can exist in both sets, and as long as at least one problem is unique to either P or NP, then $P \neq \mathrm{NP}$. Notice that some problems may still exist in both P and NP.

3. What are quantum computers, and could they efficiently solve NP-hard problems once fully developed?

Quantum computers operate on the basis of quantum mechanics. Based on superposition, entanglement, decoherence, and interference, quantum computers can encode more data at once by redefining the numerical system used by the computer: instead of using binary bits, quantum bits (qubits) are used. These qubits act like subatomic particles, though made of atoms, and can behave just like binary bits (store either `0` or `1`). However, they can also be a weighted combination of `0` and `1` at the same time. When combined, qubits can scale exponentially [4].

It is unknown if quantum computers could solve NP-hard problems efficiently once fully developed. They do show promise with quantum algorithms like Shor's algorithm, which can factor an n-bit composite number N with time complexity $O\left((\log N)^3\right)$, and Grover's algorithm, which can search through a list of n items in time $O(\sqrt{n})$ [5], [6]. As mentioned, these are promising results, but in order for NP-hard problems to be solvable by quantum computers, an exponential time reduction would be required, since as the size of an NP-hard problem increases, the time to find an optimal solution grows exponentially [7]. Grover's algorithm, for example, only provides a polynomial time reduction. Thus, more work is needed to determine if quantum computers could solve NP-hard problems once fully developed.

4. How might quantum computers affect the RSA cryptosystem? Are there alternative cryptographic systems designed to mitigate the potential threat posed by quantum computers?

As mentioned above, there is a quantum algorithm that factors an n-bit composite number N in polynomial time—Shor's algorithm (complexity $O\left((\log N)^3\right)$). This poses a real threat to RSA—if composites can be factored quickly, then RSA's power will wein, since RSA's cryptographic strength comes from how difficult it is to factor numbers. It is probable that RSA will be phased out by the time quantum computers are fully developed, in favor of other systems, under the collective umbrella of **post-quantum cryptography**. There are a number of possible directions that post quantum cryptography could take [8]:

- **Lattice-Based Cryptography**: A lattice is a set of points in some n-dimensional space with a periodic structure. The reason these could be used is that lattice problems have seen much research in the quantum domain, yet no quantum algorithm that significantly outperforms any classical algorithm are known for lattice problems. The classic problem for this branch is the shortest vector problem (SVP).
- **Hash-Based Cryptography**: Hash-based cryptosystems use hash functions to encrypt data. The simplest hash-based cryptosystem is the Merkle signature scheme, which converts from weak signature schemes to strong ones.

5. Demonstrate that if the decision version of the Hamiltonian cycle problem can be solved in polynomial time, then the task of actually finding a Hamiltonian cycle (or determining that none exists) can also be accomplished in polynomial time.

Let `HamiltonianExists(G)` be a polynomial-time algorithm that determines if G (an undirected graph) contains a Hamiltonian cycle. It returns a `boolean` value. To show that the task of actually finding a Hamiltonian cycle can be accomplished in polynomial time, an algorithm is proposed that uses `HamiltonianExists(G)`:

```
function FindHamiltonian(G):
  cycle = []
  graph = G
  for each edge e (u, v) in G:
    G' = graph \ {e}
    if not HamiltonianExists(G'):
      # if removing this edge breaks the cycle, it
      # must be part of the cycle.
      add e to cycle
    else:
      # we can remove e from the graph
      graph = G'
  return cycle
```

Notice that the decision algorithm is called $O(|E|)$ times, a linear complexity. If `HamiltonianExists(G)` is polynomial-time, then `FindHamiltonian(G)` is $O(|E| \cdot p)$, where p is some polynomial, which means `FindHamilton(G)` is a polynomial-time algorithm, since $|E| \cdot p$ is a polynomial.

Bibliography

[1] L. Fortnow, “Fifty Years of P vs. NP and the Possibility of the Impossible.” Accessed: Dec. 08, 2024. [Online]. Available: https://cacm.acm.org/research/fifty-years-of-p-vs-np-and-the-possibility-of-the-impossible/

[2] Clay Mathematics Institute, “P vs. NP.” Accessed: Dec. 08, 2024. [Online]. Available: https://www.claymath.org/millennium/p-vs-np/

[3] J. López, “RSA: How Maths Will Protect Us While P!=NP.” Accessed: Dec. 08, 2024. [Online]. Available: https://medium.com/datio-big-data/rsa-how-maths-will-protect-us-while-p-np-1b29ca6bff82

[4] J. Schneider and I. Smalley, “What is quantum computing?.” Accessed: Dec. 08, 2024. [Online]. Available: https://www.ibm.com/topics/quantum-computing

[5] M. M. Younis, A. Salim Jamil, A. H. Abdulrazzaq, N. Ahmed Mawla, R. M. Khudhair, and Y. Vasiliu, “Progress and Challenges in Quantum Computing Algorithms for NP-Hard Problems,” in *2024 36th Conference of Open Innovations Association (FRUCT)*, 2024, pp. 460–468. doi: 10.23919/FRUCT64283.2024.10749878.

[6] “Theory of Grover's search algorithm.” Accessed: Dec. 08, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/quantum/concepts-grovers

[7] B. C. B. Symons, D. Galvin, E. Sahin, V. Alexandrov, and S. Mensa, “A Practitioner's Guide to Quantum Algorithms for Optimisation Problems.” [Online]. Available: https://arxiv.org/abs/2305.07323

[8] S. Agrawal, “Post Quantum Cryptography: An Introduction.” Accessed: Dec. 08, 2024. [Online]. Available: https://cse.iitm.ac.in/~shwetaag/papers/PQC.pdf