1 \( \text{BPP} \subseteq \text{BQP} \)

To extend our earlier result that \( \text{BQP} \) simulates \( \text{P} \) to the class \( \text{BPP} \), we shall need the following 1-qubit quantum operation.

**Hadamard gate:** mapping \( |0\rangle \rightarrow |0\rangle + |1\rangle \) and \( |1\rangle \rightarrow |0\rangle - |1\rangle \). Note that applying the Hadamard operation \( n \) times (qubit by qubit) to the quantum state \( |0^n\rangle \) results in the quantum state:

\[
\sum_{x \in \{0,1\}^n} |x\rangle,
\]

the uniform state (i.e., the superposition of all possible \( 2^n \) states). If we measure this uniform quantum state, we get a classical uniformly random \( n \)-bit string! Thus, it intuitively makes sense that quantum algorithms should be able to simulate classical randomized algorithms. We make this connection precise below.

**Theorem 1.** \( \text{BPP} \subseteq \text{BQP} \).

**Proof.** Let \( L \in \text{BPP} \) be an arbitrary language. Suppose that \( R(x,y) \) is a polytime polybalanced predicate such that, for all \( x \in \{0,1\}^n \),

\[
\begin{align*}
x \in L & \Rightarrow \Pr_y[R(x,y) = 1] > 2/3, \\
x \notin L & \Rightarrow \Pr_y[R(x,y) = 0] > 2/3,
\end{align*}
\]

which exists by the definition of \( \text{BPP} \). Let \( |y| = m \), for some \( m = \text{poly}(n) \).

Consider the following quantum algorithm:

1. For a given input string \( x \), create a quantum state \( |x\rangle|0^m\rangle|0^s\rangle \), for some \( s = \text{poly}(n) \).

2. Apply the Hadamard operation \( m \) times to the second \( m \) qubits of the quantum register, getting the quantum state (up to normalization):

\[
\sum_{y \in \{0,1\}^m} |x\rangle|y\rangle|0^s\rangle.
\]
3. Next, compute $R$ on the first $n + m$ qubits of the register (using the last $s$ qubits for scratch work, as explained earlier), getting:

$$\sum_{y \in \{0, 1\}^m} |x\rangle |y\rangle |R(x, y)\rangle,$$

where we ignored the remaining scratch-work qubits.

4. Finally, measure the quantum register, getting the classical string $xyR(x, y)$, and output the last bit $R(x, y)$.

For the analysis, note that the measurement yields the state $xyR(x, y)$ for a uniformly random $y$. Hence, by definition of $L$ in terms of $R$, we get that $R(x, y)$ is correct with probability at least $2/3$. Thus, outputting the last bit, our quantum algorithm outputs the correct answer with probability $2/3$, as required.

### 1.1 Universal set of quantum operations

We finish this section by stating the following result showing that for BQP, all one needs are the Hadamard and the Toffoli operations! (And so, in particular, one does not need to use complex numbers.)

**Theorem 2** (Shi). *For the purpose of quantum computing, the Hadamard gate and the Toffoli gate form a universal set of operations. (That is, every BQP algorithm can be efficiently realized using just these two gates.)*

### 2 Quantum algorithms

There are basically three interesting quantum algorithms that seem to do better than any known classical algorithm:

- Shor’s factoring algorithm is the most famous algorithm in quantum computing, which generated a lot of research in the area. The algorithm factors a given integer number into prime factors, in time polynomial in the input bit-length. No known classical algorithm comes even close!

- Simon’s algorithm for the following “Period Search Problem”: Given a polysize classical circuit computing $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that there exists $a \in \{0, 1\}^n$, $a \neq 0^n$, with the property:

$$\forall x \neq y \in \{0, 1\}^n \quad [f(x) = f(y) \iff x = y \oplus a],$$

find this “period” string $a$. Simon’s algorithm finds $a$ in quantum polytime.
Grover’s search algorithm: Given a classical polysize circuit for \( f : \{0,1\}^n \rightarrow \{0,1\} \) such that there exists unique \( a \in \{0,1\}^n \) where \( f(a) = 1 \), find this \( a \). Grover’s algorithm finds \( a \) in time \( \text{poly}(n) \cdot 2^{n/2} \). It is still exponential, but better than the trivial \( 2^n \) time.

Shor’s factoring algorithm is rather involved to be presented in this course. So we focus on Simon’s and Grover’s algorithms (to be presented next). Simon’s algorithm is for the problem that is very similar to the subproblem one needs to solve inside Shor’s factoring algorithm. So an algorithm for Simon’s problem will give us some taste of what is involved in Shor’s algorithm.

3 Simon’s algorithm

Given a polysize classical circuit computing a function \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) such that there is a nonzero \( a \in \{0,1\}^n \) satisfying the property:

\[
\forall x \neq y \in \{0,1\}^n \quad [f(x) = f(y) \iff x = y \oplus a],
\]

we want to find this \( a \).

It’s not clear how to solve this problem classically better than by a “brute-force” exponential-time algorithm. However, it is solved in \( \text{BQP} \) by the following quantum algorithm:

1. Apply the Hadamard operation to each of the \( n \) qubits of the quantum register initialized to \( |0^n\rangle \), getting the quantum state \( \sum_{x \in \{0,1\}^n} |x\rangle \).

2. Using additional qubits initially \( |0\rangle \)'s, compute the function \( f \) of the first \( n \) qubits of the register, getting

\[
\sum_{x \in \{0,1\}^n} |x\rangle \cdot |f(x)\rangle = \sum_{x \in \{0,1\}^n} (|x\rangle + |x \oplus a\rangle) \cdot |f(x)\rangle,
\]

where the last expression is simply a rearrangement of the first using the properties of \( f \) (i.e., that \( f(x) = f(y) \) iff \( x = y \oplus a \)).

3. Measure the second \( n \) qubits, getting the quantum state \( |x\rangle + |x \oplus a\rangle \), for a uniformly random string \( x \in \{0,1\}^n \).

4. Apply Hadamard operation to each of the \( n \) qubits in the state, getting the new state:

\[
\sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} \cdot |y\rangle + \sum_{y \in \{0,1\}^n} (-1)^{x \oplus a \cdot y} \cdot |y\rangle,
\]

\[\text{Here we use the fact that applying the Hadamard operation to each qubit of the state } |x\rangle \text{ for } x = x_1 \ldots x_n \in \{0,1\}^n \text{ is } (|0\rangle + (-1)^{x_1}|1\rangle) \cdots (|0\rangle + (-1)^{x_n}|1\rangle) = \sum_{y \in \{0,1\}^n} (-1)^{\sum_{i=1}^n x_i y_i} \cdot |y\rangle.\]
where \( \langle x, y \rangle = \sum_{i=1}^{n} x_i \cdot y_i \) is the inner product of the vectors \( x \) and \( y \). The latter can be equivalently written as

\[
\sum_{y} \left( (-1)^{\langle x, y \rangle} + (-1)^{\langle x, y \rangle} \cdot (-1)^{\langle a, y \rangle} \right) \cdot |y\rangle.
\]

Note that for \( y \) such that \( \langle a, y \rangle = 1 \mod 2 \), the amplitude for \( |y\rangle \) is 0. Thus, the above quantum state is the uniform superposition over \( |y\rangle \)'s such that \( \langle a, y \rangle = 0 \mod 2 \).

5. Measure the quantum register, getting a uniformly random \( y \) such that \( \langle a, y \rangle = 0 \mod 2 \). This is a linear equation \( \mod 2 \) in the unknown \( a = a_1, \ldots, a_n \in \{0,1\}^n \), with the coefficients \( y = y_1 \ldots y_n \in \{0,1\}^n \).

6. Repeat the whole algorithm \( 2^n \) times, getting \( 2^n \) linear equations \( \mod 2 \) for the unknowns \( a \). Pick \( n-1 \) linearly independent equations from this collection, and solve for the unique value of \( a \). (It is not hard to show that once we sample enough independent random \( y \)’s such that \( \langle a, y \rangle = 0 \), we get with high probability \( n-1 \) linearly independent \( y \)'s in that collection.)

Observe that the running time of the described algorithm is polynomial in \( n \), and it finds the requisite \( a \) with high probability.

4. Grover’s search algorithm

Given a classical polynomial circuit computing \( f : \{0,1\}^n \rightarrow \{0,1\} \) such that there is unique \( a \in \{0,1\}^n \) with \( f(a) = 1 \), find this \( a \). There is a quantum algorithm that solves this problem in time \( \text{poly}(n) \cdot 2^{n/2} \). The algorithm is as follows:

1. Applying the Hadamard operation to each of the \( n \) qubits of the initially zero quantum register, get the uniform quantum state \( u = \frac{1}{2^{n/2}} \cdot \sum_{x \in \{0,1\}^n} |x\rangle \). Note that \( |a\rangle \) and \( u \) are at the angle \( \alpha \) where \( \cos \alpha = \langle u, |a\rangle \rangle = \frac{1}{2^{n/2}} \).

2. Define the vector \( e = \sum_{x \neq a} |x\rangle \), which is orthogonal to \( |a\rangle \), and is lying in the same plane as the vectors \( |a\rangle \) and \( u \).

3. Denote by \( \theta \) the angle between \( u \) and \( e \). We have \( \alpha = \pi/2 - \theta \), and so \( \cos \alpha = \sin \theta = 2^{-n/2} \). The latter implies that \( \theta \approx 2^{-n/2} \).

4. If we reflect \( u \) around \( e \), and then reflect the result around \( u \), we get a new vector \( w \) that is the angle \( 2\theta \) closer to \( |a\rangle \). We show later that such reflections can be implemented by quantum operations.

5. By repeatedly applying the reflection around \( e \) and then around \( u \) to our new vector \( w \) for about \( 2^{n/2} \) times, we will move \( w \) to within a tiny angle (say less than \( \pi/4 \)) of \( |a\rangle \). At that point, if we measure our quantum register, we get \( a \) with probability at least \( (\cos \pi/4)^2 = 1/2 \).
The above algorithm runs in about $2^{n/2}$ steps, and with high probability outputs a required $a$. We just need to argue that reflections around $e$ and $u$ can be implemented in quantum polytime. We explain this next.

**Reflection of $w$ around $e$:** Express $w$ as a linear combination of a vector collinear with $e$ and the vector orthogonal to $e$ (i.e., collinear with $|a\rangle$):

$$w = \left( \sum_{x \neq a} w_x \cdot |x\rangle \right) + w_a \cdot |a\rangle.$$  

Its reflection around $e$ is

$$w' = \left( \sum_{x \neq a} w_x \cdot |x\rangle \right) - w_a \cdot |a\rangle.$$ 

To flip the sign in the quantum state $w$ for the basis vector $|a\rangle$ (which is unknown to us!), we use the fact that $f(a) = 1$ and $f(x) = 0$ for all $x \neq a$. Using extra quantum bits (initially zero), we compute $f$ on $w$, getting the new state:

$$\sum_{x \neq a} w_x \cdot |x\rangle |0\rangle + w_a \cdot |a\rangle |1\rangle.$$ 

Then we apply the Z operation to the last qubit (mapping $|0\rangle \rightarrow |0\rangle$ and $|1\rangle \rightarrow -|1\rangle$), getting:

$$\sum_{x \neq a} w_x \cdot |x\rangle |0\rangle - w_a \cdot |a\rangle |1\rangle.$$ 

This is almost what we need, except we need to clean (zero out) the last qubit. We do that by computing $f$ in reverse, thereby undoing the earlier computation of $f$ on $w$, getting:

$$\sum_{x \neq a} w_x \cdot |x\rangle |0\rangle - w_a \cdot |a\rangle |0\rangle.$$ 

The latter is $w'$ if we ignore the last qubit.

**Reflection of $w$ around $u$:** Applying the Hadamard operation to each qubit of $w$, we get $Hw$ which we then reflect around $Hu = |0^n\rangle$, where $H$ is the Hadamard matrix of dimension $2^n \times 2^n$. We reflect $Hw$ around $|0^n\rangle$ in a way similar to that for reflection around $e$ (e.g., define $g : \{0,1\}^n \rightarrow \{0,1\}$ such that $g(0^n) = 0$ and $g(x) = 1$ for all $x \neq 0^n$, and use that $g$ in place of $f$). Let $w''$ be the reflection of $Hw$ around $Hu$. Finally, apply the Hadamard operation again to each qubit, getting $Hw''$, which turns out to be the desired reflection of $w$ around $HHu = u$. (Here we used the fact that $HH = I$ as well as the fact that $H$ is a unitary matrix which preserves the inner product, and hence the angle, between vectors, i.e., for any vectors $a, b$, we have $\langle Ha, Hb \rangle = a^* H^* Hb = a^* Ib = a^* b = \langle a, b \rangle$.)

5
5 BQP ⊆ PSPACE

Earlier we saw that BPP ⊆ BQP. Next we argue

Theorem 3. BQP ⊆ PSPACE.

Proof. Note that a BQP computation is a sequence of a polynomial number of elementary quantum operations (acting on at most 3 qubits), and a measurement in the last step. To simulate such a quantum algorithm in PSPACE, it suffices to compute in PSPACE the amplitude of each state vector $|y\rangle$, for $y \in \{0,1\}^m$, where $m$ is the polynomial size of the quantum register used by the BQP algorithm. We show that each such amplitude can computed recursively in PSPACE.

For step $t$, if the algorithm applied a unitary operation $U$ acting on qubits in positions $i, j, k$, then the amplitude of the state vector $|y\rangle$ at step $t$ depends only on the amplitudes at step $t - 1$ of the 8 state vectors obtained from $y$ by setting the bits $i, j, k$ in all possible ways. We compute these 8 amplitudes recursively, re-using space. The recurrence for the space used is $S(t) \leq S(t - 1) + \text{poly}(n)$, which solves to $\text{poly}(n)$. 

One can improve this inclusion to show that BQP ⊆ P#P.