Problem Set 7

Bayard Walsh

February 2024

1

METRIC-TSP $\in NP$

Witness
To show METRIC-TSP $\in NP$, we have the witness a, which is a permutation for a given $\langle d, k\rangle$ such that $\text{cost}(a, d) \leq k$. Given a, we can verify that the overall $\text{cost}(a, d) \leq k$ by summing all the $d(a_i, a_{i+1})$ values and checking if every city appears exactly once, which will run in polynomial time for the input. If $\text{cost}(a, d) \leq k$ and a is a valid permutation (it contains every city exactly once), we have that some permutation exists for $\langle d, k\rangle$ such that $\text{cost}(a, d) \leq k$, which shows that $\langle d, k\rangle \in$ METRIC-TSP. Therefore if we have this witness a we can verify if $\langle d, k\rangle \in$ METRIC-TSP in polynomial time, meaning METRIC-TSP $\in NP$.

METRIC-TSP $\in NP - HARD$
We will do a reduction from HAMILTONIAN cycle problem. Given some undirected graph G:

$f(\langle G = (V, E)\rangle)$
$k = |V|$
for every $v \in V$ add point $x \in \mathcal{X}$ to our set of points / cities in $1, 2, \cdots, m$
for every $e = (u, v) \in E$ let $d(u, v) = 1$
label every other distance function $d(u, v) = |V| + 1$ *the distance between any two nodes without an edge in G is* $|V| + 1$

return our constructed $\langle d, k\rangle$

YES to YES
Given a Hamiltonian cycle x for G, we can find a permutation y for $\langle d, k\rangle$ where $\text{cost}(a, d) \leq k$. Given a list of edges for our Hamiltonian cycle x as $(a_1, a_2), \cdots (a_m, a_1)$, transform these edges into the permutation $y = \text{d}(a_1, a_2) \cdots \text{d}(a_m, a_1)$, which will be a sequence of distance functions following the same order as our edge sequence. Note that by definition a Hamiltonian cycle will

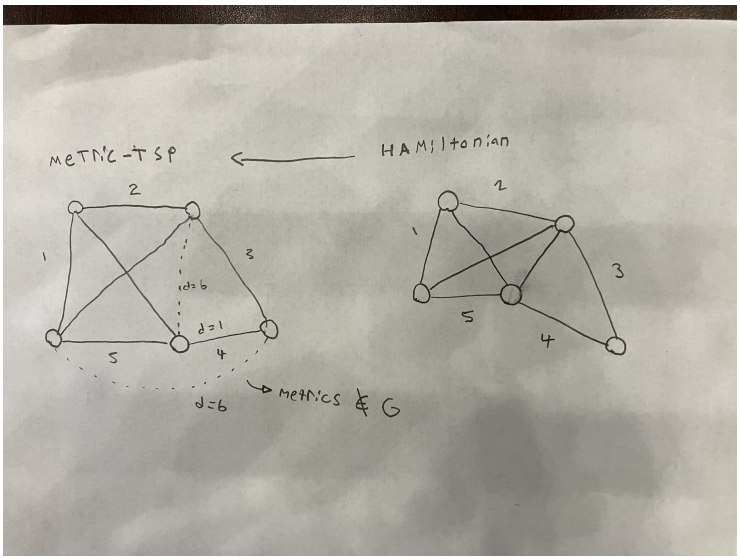

Figure 1: Drawing of reduction. Note that not all metrics $\in G$ are drawn

visit every node once and then return to the first node, so transforming the cycle into a permutation with respect to our reduction will be a witness for METRIC-TSP, as each node is transformed into a city in our reduction. Also, a Hamiltonian cycle will have $|V|$ total edges, and as each edge $\in G$ is set to $d(u, v) = 1$ in our construction of $\langle d, k \rangle$, then our overall cost function following the permutation y will be $|V|$. This means that following the same order as the Hamiltonian path, every node will appear exactly once in our permutation and we will have cost(a, d)=$|V|$, meaning cost$(a, d) \leq k$ as $|V| = k$. Therefore given a Hamiltonian path x for G, we can find a permutation y for $\langle d, k \rangle$ where cost$(a, d) \leq k$.

NO to NO

We will solve the contra positive; given a permutation y for $\langle d, k \rangle$ where cost(a, d) $\leq k$, we can find a Hamiltonian path x for G. First note that our permutation can only step between nodes that correspond to edges in G; this is because if there was any d$(a_i, a_{i+1}) \in y$ that is not an edge in G then our permutation would have a cost greater than k, each of these distance functions without an edge in G have cost $|V| + 1$, and $k = |V|$, so a single distance function outside an edge in G means the overall cost$(a, d) \not\leq k$. Next, as our permutation a must have every $x \in m$ exactly once among $a_1, a_2 \cdots a_m$, then our a will visit each city exactly once. Additionally, because cost$(a, d) \leq k$, $k = |V|$ and each $d(u, v) = 1$ for every edge in G, then the sequence of our permutation can be transformed into a Hamiltonian cycle for G consisting of exactly $|V|$ edges $\in G$. This is because each city / point in x represents a vertex in V, and our permutation will visit every city exactly once along a distance function based on the

edges in G. Therefore we can transform our permutation into a Hamiltonian cycle by taking $\mathrm{d}(a_1, a_2) \cdots \mathrm{d}(a_m, a_1)$ to be a list of edges $(a_1, a_2), \cdots (a_m, a_1)$ in a Hamiltonian cycle for G. Note that because of the nature of the traveling salesperson problem, we have a cycle as we must return to a_1 in our permutation and we can visit each city exactly once. Therefore given a permutation y for $\langle d, k \rangle$ where $\mathrm{cost}(a, d) \leq k$, we can find a Hamiltonian path x for G.

Computable
The reduction from HAMILTONIAN to METRIC-TSP involves creating a point x for every $V \in G$ and querying through every edge $(u, v) \in G$ and to assign a distance function between the points (u, v), and then assigning another distance function to every remaining possible edge $\in G$, so we have $\frac{|V|(|V|-1)}{2}$ total distance function assignments, which is polynomial. k is computed through taking count of nodes which is also polynomial. Therefore our construction of $\langle d, k \rangle$ is polynomial relative to the size of G, so our problem is computable.

2

Algorithm:
Given N
ASSERT $N \geq 2$
$p = \{\ \}$

(LOOP1)
if $N == 1$ **return** p
$L_b = 2$, $U_b = N$, $k = L_b$+floor($\frac{U_b - L_b}{2}$)

(LOOP2)
    check if FACTOR $\langle N, k \rangle$
        if yes :
            if $k == L_b$:
                concat k to p
                $N = \frac{N}{k}$
                step to *(LOOP1)*
            else:
                $U_b = k$
                $k = L_b$+floor($\frac{U_b - L_b}{2}$)
                step to *(LOOP2)*
        if no :
            if $k == L_b$:
                concat U_b to p
                $N = \frac{N}{U_b}$
                step to *(LOOP1)*
            else:
                $L_b = k$

$k = L_b$+floor($\frac{U_b - L_b}{2}$)
step to *(LOOP2)*

Correctness:
The algorithm will generate p by repeatedly dividing N by a prime factor k present in N and then updating N to be the N/k. This is done through binary searching for a viable prime factor while using the assumption that we can compute FACTOR $\langle N, k\rangle$ in polynomial time, and updating N by dividing it by a prime whenever we find one. We will repeat this until $N = 1$, meaning that when we stop iterating we have fully deconstructed N into a list of primes. We update the bounds when we accepts or reject a field of numbers as non viable primes, and use either the upper bound to divide N or the lower bound to divide N when $k = L_b$ based on if FACTOR $\langle N, k\rangle$; based on our binary search algorithm this will enable us to find a valid prime with each evaluation, meaning that each $p_i \in p$ will divide N, so the list of p will be factors that multiply up to N. We add the assertion that $N \geq 2$ as some N where $N < 2$ cannot be decomposed into prime factors.

Runtime
In our algorithm we search for viable primes using a modified version of the binary search algorithm which enables us to find a viable prime in $\log N$ loop iterations, because we halve the amount of viable numbers at each iteration. We know that there can only be at most $\log N$ prime factors for a given N, therefore we will query at most $\log N$ times to find the prime factors of N. Also as we have that integer division and FACTOR are $\in P$ by assumption, we can use these operations to find a prime in our algorithm and it will still be in polynomial time. Therefore we will check overall $O(\log N)^2$ numbers, doing polynomial work for each query, which is polynomial relative to the size of the binary encoding of N. Therefore our algorithm is polynomial.

3

As $L \in NP$, then there exists some language $R_1 \in P$, such that if $w \in L$, there exists a string x such that $|x| \leq |w|^{k_1}$ and $\langle w, x\rangle \in R_1$

As $L \in coNP$, then there exists some language $R_2 \in P$, such that if $w \notin L$, there exists a string x such that $|x| \leq |w|^{k_2}$ and $\langle w, x\rangle \in R_2$

let Γ be some alphabet that contains an encoding of $\langle w, x\rangle$ for every $w \in \Sigma^*$ and every $x \in R_1$ and $x \in R_2$. Let k be the max of k_1, k_2, with respect to $|x| \leq |w|^{k_1}$ or $|x| \leq |w|^{k_2}$. Therefore both $|x| \leq |w|^{k_1}$ or $|x| \leq |w|^{k_2}$ are bounded within k, so both are polynomial with respect to the input w.

$f : \Gamma^* \to \{0, 1, \perp\}$

$f(\langle w, x\rangle)$
    if $\langle w, x\rangle \in R_1 \rightarrow$ return 1
    if $\langle w, x\rangle \in R_2 \rightarrow$ return 0
    else return $\perp$

We can check for membership $\in R_1$ or $\in R_2$ in polynomial time as each verification language is $\in P$ so f is a polynomial-time-computable function.

Correctness
$w \in L \Rightarrow |x| \leq |w|^{k_1}$ and $f(\langle w, x\rangle) = 1$
By definition of NP, if $w \in L$, some x exists such that $|x| \leq |w|^{k_1}$ and $\langle w, x\rangle \in R_1$. If $\langle w, x\rangle \in R_1$ then $f(\langle w, x\rangle) = 1$

$w \in L \Leftarrow |x| \leq |w|^{k_1}$ and $f(\langle w, x\rangle) = 1$
If $|x| \leq |w|^{k_1}$ and $f(\langle w, x\rangle) = 1$ then $\langle w, x\rangle \in R_1$. By definition of NP, for every $w \notin L$, for every x such that $|x| \leq |w|^{k_1}$, we have $\langle w, x\rangle \notin R_1$. We take the contra positive, so if x exists such that $|x| \leq |w|^{k_1}$ and $\langle w, x\rangle \in R_1$ then $w \in L$. As we have $\langle w, x\rangle \in R_1$, we have $w \in L$.

Therefore $w \in L \Leftrightarrow |x| \leq |w|^{k_1}$ and $f(\langle w, x\rangle) = 1$

$w \notin L \Rightarrow |x| \leq |w|^{k_2}$ and $f(\langle w, x\rangle) = 0$
By definition of $coNP$, if $w \notin L$, then some x exists such that $|x| \leq |w|^{k_2}$ and $\langle w, x\rangle \in R_2$. If $\langle w, x\rangle \in R_2$ then $f(\langle w, x\rangle) = 0$

$w \notin L \Leftarrow |x| \leq |w|^{k_2}$ and $f(\langle w, x\rangle) = 0$
If $f(\langle w, x\rangle) = 0$ then $\langle w, x\rangle \in R_2$. By definition of $coNP$, for every $w \in L$, for every x such that $|x| \leq |w|^{k_2}$, we have $\langle w, x\rangle \notin R_2$. We take the contra positive, so if some x exists such that $|x| \leq |w|^{k_2}$ and $\langle w, x\rangle \in R_2$ then $w \notin L$. As we have $\langle w, x\rangle \in R_2$, we have $w \notin L$.

Therefore $w \notin L \Leftrightarrow |x| \leq |w|^{k_2}$ and $f(\langle w, x\rangle) = 0$

4

4.1

Given M_* and considering some arbitrary $w\#x$ pair, simulate M_* on w and x. In this way we can simulate M_* in polynomial-time because M_* is a polynomial-time random Turing machine, however by loading inputs w on tape 1 and x on tape 2, and accepting if M_* accepts and rejecting if M_* rejects, M_* is decidable because it either correctly accepts or rejects $w\#x$ or it incorrectly accepts or rejects $w\#x$; however either way it decides $w\#x$ for $R \in P$. As we have $P \subseteq PSIZE$, we can simulate R through a polynomial sized circuit, meaning

$R \in PSIZE$

4.2

$w \in L \Rightarrow w\#x_* \in R$
We want to show that there exists some string $x_* \in \{0,1\}^{T(n)}$ such that $w\#x_* \in R$. Given $w \in L$ for some arbitrary $w \in \{0,1\}^n$, we have that $Pr[M_*$ accepts $w] > 1 - 2^{-n}$. As we have that M_* accepts correctly $> 1 - 2^{-n}$, there is a 2^{-n} probability of error given any x_* for an arbitrary $w \in L$. We will consider $2^{-n} > \frac{|BAD_w|}{|ALL_w|}$ as our error rate for some $w \in L$, or the ratio of the size of the set of x that cause w to mistakenly not accept divided by the size of set of all possible x strings for w. As $1 > 2^{-n}$ for all $n > 0$, and $2^{-n} > \frac{|BAD_w|}{|ALL_w|}$ then $|BAD_w| < |ALL_w|$. Furthermore $BAD_w \subseteq ALL_w$, as all x that cause w to mistakenly not accept must be contained within the set of all possible x. Therefore there must exist some $x_* \in ALL_w$ and $x_* \notin BAD_w$, meaning that some x_* exists such that $Pr[M_*$ accepts $w]$ correctly when $w \in L$, so $w\#x_* \in R$.

$w\#x_* \in R \Rightarrow w \in L$
We want to show that $w \in L$. As we have $w \notin L \Rightarrow Pr[M_*$ rejects $w] > 1 - 2^{-n}$, we also have the contrapositive, that $Pr[M_*$ rejects $w] \not> 1 - 2^{-n} \Rightarrow w \in L$, or $Pr[M_*$ rejects $w] \leq 1 - 2^{-n} \Rightarrow w \in L$. Therefore we also have that $Pr[M_*$ accepts $w] \leq 2^{-n} \Rightarrow w \in L$, which represents a bound for the probability that M_* mistakenly accepts w when it should reject.

Consider if M_* could mistakenly accept w for at least 2 strings $\in x_*$. Therefore we could have $T(n) = n$, meaning that the size of overall strings for $x^* \in \{0,1\}^n$ is 2^n. Therefore we have error rate $\frac{2}{2^n}$ for this case. As we have that $\frac{2}{2^n} > \frac{1}{2^n}$, then we have a contradiction against the bound that $w \notin L \Rightarrow Pr[M_*$ rejects $w] > 1 - 2^{-n}$ for M_*, because M_* mistakenly accepts at rate $\frac{2}{2^n}$, meaning it can't reject at rate $1 - 2^{-n}$. Therefore for a given $w \notin L$, M_* cannot mistakenly accept more than one x_*.

As we have $w\#x_* \in R$, we have at least one instance where M_* accepts. Therefore considering the probability of mistakenly accepting we have $\frac{1}{2^{T(n)}}$ which is $\frac{1}{2^{T(n)}} \leq \frac{1}{2^n}$ as $T(n)$ is from naturals to naturals, meaning that $Pr[M_*$ accepts $w] \leq 2^{-n} \Rightarrow w \in L$

Therefore we have that $w \in L \Leftrightarrow w\#x_* \in R$

4.3

$L \in PSIZE$ if we can decide L with some circuit family $C = (C_0, C_1, C_2, \cdots)$ where C_n has polynomial size

Let $C = (C_0, C_1, C_2, \cdots)$ be a uniform family of poly-size circuits that decides R. For a given w we have that $f\langle w \rangle = \langle C \rangle$, where $C(x) = C_m(w\#x)$. We will show that $w \in L$ if and only if there exists x such that $C_m(w\#x) = 1$

By 4a we have that $R \in PSIZE$, so we can construct C. Given w, we can construct C_m in polynomial time, where m is the length of $w\#x_*$ in bits (w and x_* are both of polynomial length). Then we hard code $w\#$ into the circuit which takes $n^{O(1)}$ time. Therefore any $C_m \in C$ is poly computable, meaning $C \in PSIZE$.

By 4b, we have that $w \in L \Leftrightarrow w\#x_* \in R$

If $w \in L$, by 4b we have that some x_* must exist such that $w\#x_* \in R$. Since $w\#x_* \in R$ and C decides R, we have that $C_m(w\#x_*) = 1$. Therefore given $w \in L$, C decides w correctly.

If $w \notin L$, we consider the contra positive. Suppose that there exists some $C_m(w\#x_*) = 1$ for some arbitrary w. As C decides R, if $C_m(w\#x_*) = 1$ for an arbitrary w, there must exists some x_* such that $w\#x_* \in R$. By 4b if there exists some x_* such that $w\#x_* \in R$, then $w \in L$. Therefore we have the contrapositive.