Summer School on LP Solvers
Problem Set on Matrix Data Structures

Let $\tau \in \mathbb{R}^n$ be a vector with $\tau_i \geq 1$ for all i. One can show that for linear program $\min_{Ax=b} c^\top x$ the weighted log barrier

$$f_\mu(x) = c^\top x - \mu \cdot \sum_{i=1}^{n} \tau_i \cdot \log(x_i)$$

also implies a working primal-dual method. Here being centered is equivalent to

$$x_i s_i = \mu \cdot \tau_i \text{ for all } i$$

and the number of iterations is $\widetilde{O}(\sqrt{\|\tau\|_1} \cdot \log(\mu_1/\mu_0))$. (Note, this is exactly equivalent to the IPM from Monday when $\tau_i = 1$ for all i.)

For this modified IPM, rather than $\|\mathbf{S}^{-1}\delta_s\|_2, \|\mathbf{S}^{-1}\delta_x\|_2 \leq 1/10$, we have the guarantee $\|\mathbf{S}^{-1}\delta_s\|_\tau \leq 1/10$ and $\|\mathbf{X}^{-1}\delta_x\|_\tau \leq 1/10$. Here the τ-norm is defined as $\|v\|_\tau := \sqrt{\sum_{i=1}^n \tau_i \cdot v_i^2}$, i.e., the ℓ_2-norm but we weight the contribution of each component by τ_i.

1 Problem: Weighted Bound on Number of Updates

For $\epsilon \in (0, 1/2]$, let $(1-\epsilon)x \leq \overline{x} \leq (1+\epsilon)x$.

We run an algorithm (eg some IPM) that in each iteration updates $x \leftarrow x + \delta_x$ for some new δ_x. We are promised $\|\mathbf{X}^{-1}\delta_x\|_\tau \leq 1/10$. After each iteration, we update $\overline{x}_i \leftarrow x_i$ for any i where the approximation doesn't hold anymore. Let $I_k \subset \{1, ..., n\}$ be the indices i where we changed $\overline{x}_i$ during the k-th iteration.

Problem: Show that after T iterations

$$\sum_{k=1}^{T} \sum_{i \in I_k} \tau_i = O(T\sqrt{\|\tau\|_1}/\epsilon).$$

Hint: During the lecture we have proven this for $\tau = \vec{1}$, where we showed the number of updates to $\overline{x}$ is bounded by $O(T\sqrt{n}/\epsilon)$.

Remark/Application: If we have a data structure where updating $\overline{x}_i$ takes different time for different i, then we could pick τ_i to be proportional to the time. Then $\sum_{k=1}^{T} \sum_{i \in I_k} \tau_i$ would be proportional to the time complexity of our data structure. We will explore this in the next question.

2 Problem: Sparse Product Maintenance

Let $\tau_i = 1 + n \cdot \frac{\text{nnz}(\mathbf{A})_i}{\text{nnz}(\mathbf{A})}$ for all $i = 1, ..., n$. Here $\text{nnz}(\mathbf{A})_i$ is the number of non-zero entries in the ith row, and $\text{nnz}(\mathbf{A})$ is the number of non-zero entries in the entire matrix. Let us assume matrix $\mathbf{A}$ is given as a list of non-zero entries (i.e., for each row we have a list of non-zero entries).

Problem: Show we can maintain $(\mathbf{A}^\top \overline{\mathbf{X}\mathbf{S}}^{-1}\mathbf{A})$ in $O(d\,\text{nnz}(A))$ total time over $T = O(\sqrt{\|\tau\|_1})$ iterations. (In each iteration, we are promised $\|\mathbf{X}^{-1}\delta_x\|_\tau, \|\mathbf{S}^{-1}\delta_s\|_\tau \leq 1/10$.)

Remark: If we maintain $(\mathbf{A}\overline{\mathbf{X}\mathbf{S}}^{-1}\mathbf{A})$, then we compute $(\mathbf{A}\overline{\mathbf{X}\mathbf{S}}^{-1}\mathbf{A})^{-1}$ from scratch in $O(d^\omega)$ time. This implies an LP solver with $\widetilde{O}(((\sqrt{n}+d)\,\mathrm{nnz}(\mathbf{A}) + \sqrt{n}d^\omega)\log(\mu_1/\mu_0))$ time complexity. For $n > d^\omega$, this is $\widetilde{O}(\sqrt{n}\,\mathrm{nnz}(\mathbf{A})\log(\mu_1/\mu_0))$.

This is faster than the algorithms from the lecture for very sparse $\mathbf{A}$.

Extra Problem: Instead of computing the inverse from scratch in each iteration, how fast can we maintain it?