/\* ----------------------------------------------------------------------------

\* This file was automatically generated by SWIG (http://www.swig.org).

\* Version 4.0.2

\*

\* Do not make changes to this file unless you know what you are doing--modify

\* the SWIG interface file instead.

\* ----------------------------------------------------------------------------- \*/

package com.twitter.ann.faiss;

public class swigfaiss implements swigfaissConstants {

public static void bitvec\_print(SWIGTYPE\_p\_unsigned\_char b, long d) {

swigfaissJNI.bitvec\_print(SWIGTYPE\_p\_unsigned\_char.getCPtr(b), d);

}

public static void fvecs2bitvecs(SWIGTYPE\_p\_float x, SWIGTYPE\_p\_unsigned\_char b, long d, long n) {

swigfaissJNI.fvecs2bitvecs(SWIGTYPE\_p\_float.getCPtr(x), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), d, n);

}

public static void bitvecs2fvecs(SWIGTYPE\_p\_unsigned\_char b, SWIGTYPE\_p\_float x, long d, long n) {

swigfaissJNI.bitvecs2fvecs(SWIGTYPE\_p\_unsigned\_char.getCPtr(b), SWIGTYPE\_p\_float.getCPtr(x), d, n);

}

public static void fvec2bitvec(SWIGTYPE\_p\_float x, SWIGTYPE\_p\_unsigned\_char b, long d) {

swigfaissJNI.fvec2bitvec(SWIGTYPE\_p\_float.getCPtr(x), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), d);

}

public static void bitvec\_shuffle(long n, long da, long db, SWIGTYPE\_p\_int order, SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b) {

swigfaissJNI.bitvec\_shuffle(n, da, db, SWIGTYPE\_p\_int.getCPtr(order), SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b));

}

public static void setHamming\_batch\_size(long value) {

swigfaissJNI.hamming\_batch\_size\_set(value);

}

public static long getHamming\_batch\_size() {

return swigfaissJNI.hamming\_batch\_size\_get();

}

public static int popcount64(long x) {

return swigfaissJNI.popcount64(x);

}

public static void hammings(SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long na, long nb, long nbytespercode, SWIGTYPE\_p\_int dis) {

swigfaissJNI.hammings(SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), na, nb, nbytespercode, SWIGTYPE\_p\_int.getCPtr(dis));

}

public static void hammings\_knn\_hc(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t ha, SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long nb, long ncodes, int ordered) {

swigfaissJNI.hammings\_knn\_hc(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t.getCPtr(ha), SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), nb, ncodes, ordered);

}

public static void hammings\_knn(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t ha, SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long nb, long ncodes, int ordered) {

swigfaissJNI.hammings\_knn(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t.getCPtr(ha), SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), nb, ncodes, ordered);

}

public static void hammings\_knn\_mc(SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long na, long nb, long k, long ncodes, SWIGTYPE\_p\_int distances, LongVector labels) {

swigfaissJNI.hammings\_knn\_mc(SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), na, nb, k, ncodes, SWIGTYPE\_p\_int.getCPtr(distances), SWIGTYPE\_p\_long\_long.getCPtr(labels.data()), labels);

}

public static void hamming\_range\_search(SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long na, long nb, int radius, long ncodes, RangeSearchResult result) {

swigfaissJNI.hamming\_range\_search(SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), na, nb, radius, ncodes, RangeSearchResult.getCPtr(result), result);

}

public static void hamming\_count\_thres(SWIGTYPE\_p\_unsigned\_char bs1, SWIGTYPE\_p\_unsigned\_char bs2, long n1, long n2, int ht, long ncodes, SWIGTYPE\_p\_unsigned\_long nptr) {

swigfaissJNI.hamming\_count\_thres(SWIGTYPE\_p\_unsigned\_char.getCPtr(bs1), SWIGTYPE\_p\_unsigned\_char.getCPtr(bs2), n1, n2, ht, ncodes, SWIGTYPE\_p\_unsigned\_long.getCPtr(nptr));

}

public static long match\_hamming\_thres(SWIGTYPE\_p\_unsigned\_char bs1, SWIGTYPE\_p\_unsigned\_char bs2, long n1, long n2, int ht, long ncodes, LongVector idx, SWIGTYPE\_p\_int dis) {

return swigfaissJNI.match\_hamming\_thres(SWIGTYPE\_p\_unsigned\_char.getCPtr(bs1), SWIGTYPE\_p\_unsigned\_char.getCPtr(bs2), n1, n2, ht, ncodes, SWIGTYPE\_p\_long\_long.getCPtr(idx.data()), idx, SWIGTYPE\_p\_int.getCPtr(dis));

}

public static void crosshamming\_count\_thres(SWIGTYPE\_p\_unsigned\_char dbs, long n, int ht, long ncodes, SWIGTYPE\_p\_unsigned\_long nptr) {

swigfaissJNI.crosshamming\_count\_thres(SWIGTYPE\_p\_unsigned\_char.getCPtr(dbs), n, ht, ncodes, SWIGTYPE\_p\_unsigned\_long.getCPtr(nptr));

}

public static int get\_num\_gpus() {

return swigfaissJNI.get\_num\_gpus();

}

public static String get\_compile\_options() {

return swigfaissJNI.get\_compile\_options();

}

public static double getmillisecs() {

return swigfaissJNI.getmillisecs();

}

public static long get\_mem\_usage\_kb() {

return swigfaissJNI.get\_mem\_usage\_kb();

}

public static long get\_cycles() {

return swigfaissJNI.get\_cycles();

}

public static void fvec\_madd(long n, SWIGTYPE\_p\_float a, float bf, SWIGTYPE\_p\_float b, SWIGTYPE\_p\_float c) {

swigfaissJNI.fvec\_madd(n, SWIGTYPE\_p\_float.getCPtr(a), bf, SWIGTYPE\_p\_float.getCPtr(b), SWIGTYPE\_p\_float.getCPtr(c));

}

public static int fvec\_madd\_and\_argmin(long n, SWIGTYPE\_p\_float a, float bf, SWIGTYPE\_p\_float b, SWIGTYPE\_p\_float c) {

return swigfaissJNI.fvec\_madd\_and\_argmin(n, SWIGTYPE\_p\_float.getCPtr(a), bf, SWIGTYPE\_p\_float.getCPtr(b), SWIGTYPE\_p\_float.getCPtr(c));

}

public static void reflection(SWIGTYPE\_p\_float u, SWIGTYPE\_p\_float x, long n, long d, long nu) {

swigfaissJNI.reflection(SWIGTYPE\_p\_float.getCPtr(u), SWIGTYPE\_p\_float.getCPtr(x), n, d, nu);

}

public static void matrix\_qr(int m, int n, SWIGTYPE\_p\_float a) {

swigfaissJNI.matrix\_qr(m, n, SWIGTYPE\_p\_float.getCPtr(a));

}

public static void ranklist\_handle\_ties(int k, LongVector idx, SWIGTYPE\_p\_float dis) {

swigfaissJNI.ranklist\_handle\_ties(k, SWIGTYPE\_p\_long\_long.getCPtr(idx.data()), idx, SWIGTYPE\_p\_float.getCPtr(dis));

}

public static long ranklist\_intersection\_size(long k1, LongVector v1, long k2, LongVector v2) {

return swigfaissJNI.ranklist\_intersection\_size(k1, SWIGTYPE\_p\_long\_long.getCPtr(v1.data()), v1, k2, SWIGTYPE\_p\_long\_long.getCPtr(v2.data()), v2);

}

public static long merge\_result\_table\_with(long n, long k, LongVector I0, SWIGTYPE\_p\_float D0, LongVector I1, SWIGTYPE\_p\_float D1, boolean keep\_min, long translation) {

return swigfaissJNI.merge\_result\_table\_with\_\_SWIG\_0(n, k, SWIGTYPE\_p\_long\_long.getCPtr(I0.data()), I0, SWIGTYPE\_p\_float.getCPtr(D0), SWIGTYPE\_p\_long\_long.getCPtr(I1.data()), I1, SWIGTYPE\_p\_float.getCPtr(D1), keep\_min, translation);

}

public static long merge\_result\_table\_with(long n, long k, LongVector I0, SWIGTYPE\_p\_float D0, LongVector I1, SWIGTYPE\_p\_float D1, boolean keep\_min) {

return swigfaissJNI.merge\_result\_table\_with\_\_SWIG\_1(n, k, SWIGTYPE\_p\_long\_long.getCPtr(I0.data()), I0, SWIGTYPE\_p\_float.getCPtr(D0), SWIGTYPE\_p\_long\_long.getCPtr(I1.data()), I1, SWIGTYPE\_p\_float.getCPtr(D1), keep\_min);

}

public static long merge\_result\_table\_with(long n, long k, LongVector I0, SWIGTYPE\_p\_float D0, LongVector I1, SWIGTYPE\_p\_float D1) {

return swigfaissJNI.merge\_result\_table\_with\_\_SWIG\_2(n, k, SWIGTYPE\_p\_long\_long.getCPtr(I0.data()), I0, SWIGTYPE\_p\_float.getCPtr(D0), SWIGTYPE\_p\_long\_long.getCPtr(I1.data()), I1, SWIGTYPE\_p\_float.getCPtr(D1));

}

public static double imbalance\_factor(int n, int k, LongVector assign) {

return swigfaissJNI.imbalance\_factor\_\_SWIG\_0(n, k, SWIGTYPE\_p\_long\_long.getCPtr(assign.data()), assign);

}

public static double imbalance\_factor(int k, SWIGTYPE\_p\_int hist) {

return swigfaissJNI.imbalance\_factor\_\_SWIG\_1(k, SWIGTYPE\_p\_int.getCPtr(hist));

}

public static void fvec\_argsort(long n, SWIGTYPE\_p\_float vals, SWIGTYPE\_p\_unsigned\_long perm) {

swigfaissJNI.fvec\_argsort(n, SWIGTYPE\_p\_float.getCPtr(vals), SWIGTYPE\_p\_unsigned\_long.getCPtr(perm));

}

public static void fvec\_argsort\_parallel(long n, SWIGTYPE\_p\_float vals, SWIGTYPE\_p\_unsigned\_long perm) {

swigfaissJNI.fvec\_argsort\_parallel(n, SWIGTYPE\_p\_float.getCPtr(vals), SWIGTYPE\_p\_unsigned\_long.getCPtr(perm));

}

public static int ivec\_hist(long n, SWIGTYPE\_p\_int v, int vmax, SWIGTYPE\_p\_int hist) {

return swigfaissJNI.ivec\_hist(n, SWIGTYPE\_p\_int.getCPtr(v), vmax, SWIGTYPE\_p\_int.getCPtr(hist));

}

public static void bincode\_hist(long n, long nbits, SWIGTYPE\_p\_unsigned\_char codes, SWIGTYPE\_p\_int hist) {

swigfaissJNI.bincode\_hist(n, nbits, SWIGTYPE\_p\_unsigned\_char.getCPtr(codes), SWIGTYPE\_p\_int.getCPtr(hist));

}

public static long ivec\_checksum(long n, SWIGTYPE\_p\_int a) {

return swigfaissJNI.ivec\_checksum(n, SWIGTYPE\_p\_int.getCPtr(a));

}

public static SWIGTYPE\_p\_float fvecs\_maybe\_subsample(long d, SWIGTYPE\_p\_unsigned\_long n, long nmax, SWIGTYPE\_p\_float x, boolean verbose, long seed) {

long cPtr = swigfaissJNI.fvecs\_maybe\_subsample\_\_SWIG\_0(d, SWIGTYPE\_p\_unsigned\_long.getCPtr(n), nmax, SWIGTYPE\_p\_float.getCPtr(x), verbose, seed);

return (cPtr == 0) ? null : new SWIGTYPE\_p\_float(cPtr, false);

}

public static SWIGTYPE\_p\_float fvecs\_maybe\_subsample(long d, SWIGTYPE\_p\_unsigned\_long n, long nmax, SWIGTYPE\_p\_float x, boolean verbose) {

long cPtr = swigfaissJNI.fvecs\_maybe\_subsample\_\_SWIG\_1(d, SWIGTYPE\_p\_unsigned\_long.getCPtr(n), nmax, SWIGTYPE\_p\_float.getCPtr(x), verbose);

return (cPtr == 0) ? null : new SWIGTYPE\_p\_float(cPtr, false);

}

public static SWIGTYPE\_p\_float fvecs\_maybe\_subsample(long d, SWIGTYPE\_p\_unsigned\_long n, long nmax, SWIGTYPE\_p\_float x) {

long cPtr = swigfaissJNI.fvecs\_maybe\_subsample\_\_SWIG\_2(d, SWIGTYPE\_p\_unsigned\_long.getCPtr(n), nmax, SWIGTYPE\_p\_float.getCPtr(x));

return (cPtr == 0) ? null : new SWIGTYPE\_p\_float(cPtr, false);

}

public static void binary\_to\_real(long d, SWIGTYPE\_p\_unsigned\_char x\_in, SWIGTYPE\_p\_float x\_out) {

swigfaissJNI.binary\_to\_real(d, SWIGTYPE\_p\_unsigned\_char.getCPtr(x\_in), SWIGTYPE\_p\_float.getCPtr(x\_out));

}

public static void real\_to\_binary(long d, SWIGTYPE\_p\_float x\_in, SWIGTYPE\_p\_unsigned\_char x\_out) {

swigfaissJNI.real\_to\_binary(d, SWIGTYPE\_p\_float.getCPtr(x\_in), SWIGTYPE\_p\_unsigned\_char.getCPtr(x\_out));

}

public static long hash\_bytes(SWIGTYPE\_p\_unsigned\_char bytes, long n) {

return swigfaissJNI.hash\_bytes(SWIGTYPE\_p\_unsigned\_char.getCPtr(bytes), n);

}

public static boolean check\_openmp() {

return swigfaissJNI.check\_openmp();

}

public static float kmeans\_clustering(long d, long n, long k, SWIGTYPE\_p\_float x, SWIGTYPE\_p\_float centroids) {

return swigfaissJNI.kmeans\_clustering(d, n, k, SWIGTYPE\_p\_float.getCPtr(x), SWIGTYPE\_p\_float.getCPtr(centroids));

}

public static void setIndexPQ\_stats(IndexPQStats value) {

swigfaissJNI.indexPQ\_stats\_set(IndexPQStats.getCPtr(value), value);

}

public static IndexPQStats getIndexPQ\_stats() {

long cPtr = swigfaissJNI.indexPQ\_stats\_get();

return (cPtr == 0) ? null : new IndexPQStats(cPtr, false);

}

public static void setIndexIVF\_stats(IndexIVFStats value) {

swigfaissJNI.indexIVF\_stats\_set(IndexIVFStats.getCPtr(value), value);

}

public static IndexIVFStats getIndexIVF\_stats() {

long cPtr = swigfaissJNI.indexIVF\_stats\_get();

return (cPtr == 0) ? null : new IndexIVFStats(cPtr, false);

}

public static short[] getHamdis\_tab\_ham\_bytes() {

return swigfaissJNI.hamdis\_tab\_ham\_bytes\_get();

}

public static int generalized\_hamming\_64(long a) {

return swigfaissJNI.generalized\_hamming\_64(a);

}

public static void generalized\_hammings\_knn\_hc(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t ha, SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long nb, long code\_size, int ordered) {

swigfaissJNI.generalized\_hammings\_knn\_hc\_\_SWIG\_0(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t.getCPtr(ha), SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), nb, code\_size, ordered);

}

public static void generalized\_hammings\_knn\_hc(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t ha, SWIGTYPE\_p\_unsigned\_char a, SWIGTYPE\_p\_unsigned\_char b, long nb, long code\_size) {

swigfaissJNI.generalized\_hammings\_knn\_hc\_\_SWIG\_1(SWIGTYPE\_p\_faiss\_\_HeapArrayT\_faiss\_\_CMaxT\_int\_int64\_t\_t\_t.getCPtr(ha), SWIGTYPE\_p\_unsigned\_char.getCPtr(a), SWIGTYPE\_p\_unsigned\_char.getCPtr(b), nb, code\_size);

}

public static void check\_compatible\_for\_merge(Index index1, Index index2) {

swigfaissJNI.check\_compatible\_for\_merge(Index.getCPtr(index1), index1, Index.getCPtr(index2), index2);

}

public static IndexIVF extract\_index\_ivf(Index index) {

long cPtr = swigfaissJNI.extract\_index\_ivf\_\_SWIG\_0(Index.getCPtr(index), index);

return (cPtr == 0) ? null : new IndexIVF(cPtr, false);

}

public static IndexIVF try\_extract\_index\_ivf(Index index) {

long cPtr = swigfaissJNI.try\_extract\_index\_ivf\_\_SWIG\_0(Index.getCPtr(index), index);

return (cPtr == 0) ? null : new IndexIVF(cPtr, false);

}

public static void merge\_into(Index index0, Index index1, boolean shift\_ids) {

swigfaissJNI.merge\_into(Index.getCPtr(index0), index0, Index.getCPtr(index1), index1, shift\_ids);

}

public static void search\_centroid(Index index, SWIGTYPE\_p\_float x, int n, LongVector centroid\_ids) {

swigfaissJNI.search\_centroid(Index.getCPtr(index), index, SWIGTYPE\_p\_float.getCPtr(x), n, SWIGTYPE\_p\_long\_long.getCPtr(centroid\_ids.data()), centroid\_ids);

}

public static void search\_and\_return\_centroids(Index index, long n, SWIGTYPE\_p\_float xin, int k, SWIGTYPE\_p\_float distances, LongVector labels, LongVector query\_centroid\_ids, LongVector result\_centroid\_ids) {

swigfaissJNI.search\_and\_return\_centroids(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(xin), k, SWIGTYPE\_p\_float.getCPtr(distances), SWIGTYPE\_p\_long\_long.getCPtr(labels.data()), labels, SWIGTYPE\_p\_long\_long.getCPtr(query\_centroid\_ids.data()), query\_centroid\_ids, SWIGTYPE\_p\_long\_long.getCPtr(result\_centroid\_ids.data()), result\_centroid\_ids);

}

public static ArrayInvertedLists get\_invlist\_range(Index index, int i0, int i1) {

long cPtr = swigfaissJNI.get\_invlist\_range(Index.getCPtr(index), index, i0, i1);

return (cPtr == 0) ? null : new ArrayInvertedLists(cPtr, false);

}

public static void set\_invlist\_range(Index index, int i0, int i1, ArrayInvertedLists src) {

swigfaissJNI.set\_invlist\_range(Index.getCPtr(index), index, i0, i1, ArrayInvertedLists.getCPtr(src), src);

}

public static void search\_with\_parameters(Index index, long n, SWIGTYPE\_p\_float x, long k, SWIGTYPE\_p\_float distances, LongVector labels, IVFSearchParameters params, SWIGTYPE\_p\_unsigned\_long nb\_dis, SWIGTYPE\_p\_double ms\_per\_stage) {

swigfaissJNI.search\_with\_parameters\_\_SWIG\_0(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(x), k, SWIGTYPE\_p\_float.getCPtr(distances), SWIGTYPE\_p\_long\_long.getCPtr(labels.data()), labels, IVFSearchParameters.getCPtr(params), params, SWIGTYPE\_p\_unsigned\_long.getCPtr(nb\_dis), SWIGTYPE\_p\_double.getCPtr(ms\_per\_stage));

}

public static void search\_with\_parameters(Index index, long n, SWIGTYPE\_p\_float x, long k, SWIGTYPE\_p\_float distances, LongVector labels, IVFSearchParameters params, SWIGTYPE\_p\_unsigned\_long nb\_dis) {

swigfaissJNI.search\_with\_parameters\_\_SWIG\_1(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(x), k, SWIGTYPE\_p\_float.getCPtr(distances), SWIGTYPE\_p\_long\_long.getCPtr(labels.data()), labels, IVFSearchParameters.getCPtr(params), params, SWIGTYPE\_p\_unsigned\_long.getCPtr(nb\_dis));

}

public static void search\_with\_parameters(Index index, long n, SWIGTYPE\_p\_float x, long k, SWIGTYPE\_p\_float distances, LongVector labels, IVFSearchParameters params) {

swigfaissJNI.search\_with\_parameters\_\_SWIG\_2(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(x), k, SWIGTYPE\_p\_float.getCPtr(distances), SWIGTYPE\_p\_long\_long.getCPtr(labels.data()), labels, IVFSearchParameters.getCPtr(params), params);

}

public static void range\_search\_with\_parameters(Index index, long n, SWIGTYPE\_p\_float x, float radius, RangeSearchResult result, IVFSearchParameters params, SWIGTYPE\_p\_unsigned\_long nb\_dis, SWIGTYPE\_p\_double ms\_per\_stage) {

swigfaissJNI.range\_search\_with\_parameters\_\_SWIG\_0(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(x), radius, RangeSearchResult.getCPtr(result), result, IVFSearchParameters.getCPtr(params), params, SWIGTYPE\_p\_unsigned\_long.getCPtr(nb\_dis), SWIGTYPE\_p\_double.getCPtr(ms\_per\_stage));

}

public static void range\_search\_with\_parameters(Index index, long n, SWIGTYPE\_p\_float x, float radius, RangeSearchResult result, IVFSearchParameters params, SWIGTYPE\_p\_unsigned\_long nb\_dis) {

swigfaissJNI.range\_search\_with\_parameters\_\_SWIG\_1(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(x), radius, RangeSearchResult.getCPtr(result), result, IVFSearchParameters.getCPtr(params), params, SWIGTYPE\_p\_unsigned\_long.getCPtr(nb\_dis));

}

public static void range\_search\_with\_parameters(Index index, long n, SWIGTYPE\_p\_float x, float radius, RangeSearchResult result, IVFSearchParameters params) {

swigfaissJNI.range\_search\_with\_parameters\_\_SWIG\_2(Index.getCPtr(index), index, n, SWIGTYPE\_p\_float.getCPtr(x), radius, RangeSearchResult.getCPtr(result), result, IVFSearchParameters.getCPtr(params), params);

}

public static void setHnsw\_stats(HNSWStats value) {

swigfaissJNI.hnsw\_stats\_set(HNSWStats.getCPtr(value), value);

}

public static HNSWStats getHnsw\_stats() {

long cPtr = swigfaissJNI.hnsw\_stats\_get();

return (cPtr == 0) ? null : new HNSWStats(cPtr, false);

}

public static void setPrecomputed\_table\_max\_bytes(long value) {

swigfaissJNI.precomputed\_table\_max\_bytes\_set(value);

}

public static long getPrecomputed\_table\_max\_bytes() {

return swigfaissJNI.precomputed\_table\_max\_bytes\_get();

}

public static void initialize\_IVFPQ\_precomputed\_table(SWIGTYPE\_p\_int use\_precomputed\_table, Index quantizer, ProductQuantizer pq, SWIGTYPE\_p\_AlignedTableT\_float\_32\_t precomputed\_table, boolean verbose) {

swigfaissJNI.initialize\_IVFPQ\_precomputed\_table(SWIGTYPE\_p\_int.getCPtr(use\_precomputed\_table), Index.getCPtr(quantizer), quantizer, ProductQuantizer.getCPtr(pq), pq, SWIGTYPE\_p\_AlignedTableT\_float\_32\_t.getCPtr(precomputed\_table), verbose);

}

public static void setIndexIVFPQ\_stats(IndexIVFPQStats value) {

swigfaissJNI.indexIVFPQ\_stats\_set(IndexIVFPQStats.getCPtr(value), value);

}

public static IndexIVFPQStats getIndexIVFPQ\_stats() {

long cPtr = swigfaissJNI.indexIVFPQ\_stats\_get();

return (cPtr == 0) ? null : new IndexIVFPQStats(cPtr, false);

}

public static Index downcast\_index(Index index) {

long cPtr = swigfaissJNI.downcast\_index(Index.getCPtr(index), index);

return (cPtr == 0) ? null : new Index(cPtr, false);

}

public static VectorTransform downcast\_VectorTransform(VectorTransform vt) {

long cPtr = swigfaissJNI.downcast\_VectorTransform(VectorTransform.getCPtr(vt), vt);

return (cPtr == 0) ? null : new VectorTransform(cPtr, false);

}

public static IndexBinary downcast\_IndexBinary(IndexBinary index) {

long cPtr = swigfaissJNI.downcast\_IndexBinary(IndexBinary.getCPtr(index), index);

return (cPtr == 0) ? null : new IndexBinary(cPtr, false);

}

public static Index upcast\_IndexShards(IndexShards index) {

long cPtr = swigfaissJNI.upcast\_IndexShards(IndexShards.getCPtr(index), index);

return (cPtr == 0) ? null : new Index(cPtr, false);

}

public static void write\_index(Index idx, String fname) {

swigfaissJNI.write\_index\_\_SWIG\_0(Index.getCPtr(idx), idx, fname);

}

public static void write\_index(Index idx, SWIGTYPE\_p\_FILE f) {

swigfaissJNI.write\_index\_\_SWIG\_1(Index.getCPtr(idx), idx, SWIGTYPE\_p\_FILE.getCPtr(f));

}

public static void write\_index(Index idx, SWIGTYPE\_p\_faiss\_\_IOWriter writer) {

swigfaissJNI.write\_index\_\_SWIG\_2(Index.getCPtr(idx), idx, SWIGTYPE\_p\_faiss\_\_IOWriter.getCPtr(writer));

}

public static void write\_index\_binary(IndexBinary idx, String fname) {

swigfaissJNI.write\_index\_binary\_\_SWIG\_0(IndexBinary.getCPtr(idx), idx, fname);

}

public static void write\_index\_binary(IndexBinary idx, SWIGTYPE\_p\_FILE f) {

swigfaissJNI.write\_index\_binary\_\_SWIG\_1(IndexBinary.getCPtr(idx), idx, SWIGTYPE\_p\_FILE.getCPtr(f));

}

public static void write\_index\_binary(IndexBinary idx, SWIGTYPE\_p\_faiss\_\_IOWriter writer) {

swigfaissJNI.write\_index\_binary\_\_SWIG\_2(IndexBinary.getCPtr(idx), idx, SWIGTYPE\_p\_faiss\_\_IOWriter.getCPtr(writer));

}

public static int getIO\_FLAG\_READ\_ONLY() {

return swigfaissJNI.IO\_FLAG\_READ\_ONLY\_get();

}

public static int getIO\_FLAG\_ONDISK\_SAME\_DIR() {

return swigfaissJNI.IO\_FLAG\_ONDISK\_SAME\_DIR\_get();

}

public static int getIO\_FLAG\_SKIP\_IVF\_DATA() {

return swigfaissJNI.IO\_FLAG\_SKIP\_IVF\_DATA\_get();

}

public static int getIO\_FLAG\_MMAP() {

return swigfaissJNI.IO\_FLAG\_MMAP\_get();

}

public static Index read\_index(String fname, int io\_flags) {

long cPtr = swigfaissJNI.read\_index\_\_SWIG\_0(fname, io\_flags);

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static Index read\_index(String fname) {

long cPtr = swigfaissJNI.read\_index\_\_SWIG\_1(fname);

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static Index read\_index(SWIGTYPE\_p\_FILE f, int io\_flags) {

long cPtr = swigfaissJNI.read\_index\_\_SWIG\_2(SWIGTYPE\_p\_FILE.getCPtr(f), io\_flags);

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static Index read\_index(SWIGTYPE\_p\_FILE f) {

long cPtr = swigfaissJNI.read\_index\_\_SWIG\_3(SWIGTYPE\_p\_FILE.getCPtr(f));

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static Index read\_index(SWIGTYPE\_p\_faiss\_\_IOReader reader, int io\_flags) {

long cPtr = swigfaissJNI.read\_index\_\_SWIG\_4(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader), io\_flags);

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static Index read\_index(SWIGTYPE\_p\_faiss\_\_IOReader reader) {

long cPtr = swigfaissJNI.read\_index\_\_SWIG\_5(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader));

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static IndexBinary read\_index\_binary(String fname, int io\_flags) {

long cPtr = swigfaissJNI.read\_index\_binary\_\_SWIG\_0(fname, io\_flags);

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static IndexBinary read\_index\_binary(String fname) {

long cPtr = swigfaissJNI.read\_index\_binary\_\_SWIG\_1(fname);

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static IndexBinary read\_index\_binary(SWIGTYPE\_p\_FILE f, int io\_flags) {

long cPtr = swigfaissJNI.read\_index\_binary\_\_SWIG\_2(SWIGTYPE\_p\_FILE.getCPtr(f), io\_flags);

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static IndexBinary read\_index\_binary(SWIGTYPE\_p\_FILE f) {

long cPtr = swigfaissJNI.read\_index\_binary\_\_SWIG\_3(SWIGTYPE\_p\_FILE.getCPtr(f));

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static IndexBinary read\_index\_binary(SWIGTYPE\_p\_faiss\_\_IOReader reader, int io\_flags) {

long cPtr = swigfaissJNI.read\_index\_binary\_\_SWIG\_4(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader), io\_flags);

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static IndexBinary read\_index\_binary(SWIGTYPE\_p\_faiss\_\_IOReader reader) {

long cPtr = swigfaissJNI.read\_index\_binary\_\_SWIG\_5(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader));

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static void write\_VectorTransform(VectorTransform vt, String fname) {

swigfaissJNI.write\_VectorTransform(VectorTransform.getCPtr(vt), vt, fname);

}

public static VectorTransform read\_VectorTransform(String fname) {

long cPtr = swigfaissJNI.read\_VectorTransform(fname);

return (cPtr == 0) ? null : new VectorTransform(cPtr, true);

}

public static ProductQuantizer read\_ProductQuantizer(String fname) {

long cPtr = swigfaissJNI.read\_ProductQuantizer\_\_SWIG\_0(fname);

return (cPtr == 0) ? null : new ProductQuantizer(cPtr, true);

}

public static ProductQuantizer read\_ProductQuantizer(SWIGTYPE\_p\_faiss\_\_IOReader reader) {

long cPtr = swigfaissJNI.read\_ProductQuantizer\_\_SWIG\_1(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader));

return (cPtr == 0) ? null : new ProductQuantizer(cPtr, true);

}

public static void write\_ProductQuantizer(ProductQuantizer pq, String fname) {

swigfaissJNI.write\_ProductQuantizer\_\_SWIG\_0(ProductQuantizer.getCPtr(pq), pq, fname);

}

public static void write\_ProductQuantizer(ProductQuantizer pq, SWIGTYPE\_p\_faiss\_\_IOWriter f) {

swigfaissJNI.write\_ProductQuantizer\_\_SWIG\_1(ProductQuantizer.getCPtr(pq), pq, SWIGTYPE\_p\_faiss\_\_IOWriter.getCPtr(f));

}

public static void write\_InvertedLists(InvertedLists ils, SWIGTYPE\_p\_faiss\_\_IOWriter f) {

swigfaissJNI.write\_InvertedLists(InvertedLists.getCPtr(ils), ils, SWIGTYPE\_p\_faiss\_\_IOWriter.getCPtr(f));

}

public static InvertedLists read\_InvertedLists(SWIGTYPE\_p\_faiss\_\_IOReader reader, int io\_flags) {

long cPtr = swigfaissJNI.read\_InvertedLists\_\_SWIG\_0(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader), io\_flags);

return (cPtr == 0) ? null : new InvertedLists(cPtr, false);

}

public static InvertedLists read\_InvertedLists(SWIGTYPE\_p\_faiss\_\_IOReader reader) {

long cPtr = swigfaissJNI.read\_InvertedLists\_\_SWIG\_1(SWIGTYPE\_p\_faiss\_\_IOReader.getCPtr(reader));

return (cPtr == 0) ? null : new InvertedLists(cPtr, false);

}

public static Index index\_factory(int d, String description, MetricType metric) {

long cPtr = swigfaissJNI.index\_factory\_\_SWIG\_0(d, description, metric.swigValue());

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static Index index\_factory(int d, String description) {

long cPtr = swigfaissJNI.index\_factory\_\_SWIG\_1(d, description);

return (cPtr == 0) ? null : new Index(cPtr, true);

}

public static void setIndex\_factory\_verbose(int value) {

swigfaissJNI.index\_factory\_verbose\_set(value);

}

public static int getIndex\_factory\_verbose() {

return swigfaissJNI.index\_factory\_verbose\_get();

}

public static IndexBinary index\_binary\_factory(int d, String description) {

long cPtr = swigfaissJNI.index\_binary\_factory(d, description);

return (cPtr == 0) ? null : new IndexBinary(cPtr, true);

}

public static void simd\_histogram\_8(SWIGTYPE\_p\_uint16\_t data, int n, SWIGTYPE\_p\_uint16\_t min, int shift, SWIGTYPE\_p\_int hist) {

swigfaissJNI.simd\_histogram\_8(SWIGTYPE\_p\_uint16\_t.getCPtr(data), n, SWIGTYPE\_p\_uint16\_t.getCPtr(min), shift, SWIGTYPE\_p\_int.getCPtr(hist));

}

public static void simd\_histogram\_16(SWIGTYPE\_p\_uint16\_t data, int n, SWIGTYPE\_p\_uint16\_t min, int shift, SWIGTYPE\_p\_int hist) {

swigfaissJNI.simd\_histogram\_16(SWIGTYPE\_p\_uint16\_t.getCPtr(data), n, SWIGTYPE\_p\_uint16\_t.getCPtr(min), shift, SWIGTYPE\_p\_int.getCPtr(hist));

}

public static void setPartition\_stats(PartitionStats value) {

swigfaissJNI.partition\_stats\_set(PartitionStats.getCPtr(value), value);

}

public static PartitionStats getPartition\_stats() {

long cPtr = swigfaissJNI.partition\_stats\_get();

return (cPtr == 0) ? null : new PartitionStats(cPtr, false);

}

public static float CMin\_float\_partition\_fuzzy(SWIGTYPE\_p\_float vals, LongVector ids, long n, long q\_min, long q\_max, SWIGTYPE\_p\_unsigned\_long q\_out) {

return swigfaissJNI.CMin\_float\_partition\_fuzzy(SWIGTYPE\_p\_float.getCPtr(vals), SWIGTYPE\_p\_long\_long.getCPtr(ids.data()), ids, n, q\_min, q\_max, SWIGTYPE\_p\_unsigned\_long.getCPtr(q\_out));

}

public static float CMax\_float\_partition\_fuzzy(SWIGTYPE\_p\_float vals, LongVector ids, long n, long q\_min, long q\_max, SWIGTYPE\_p\_unsigned\_long q\_out) {

return swigfaissJNI.CMax\_float\_partition\_fuzzy(SWIGTYPE\_p\_float.getCPtr(vals), SWIGTYPE\_p\_long\_long.getCPtr(ids.data()), ids, n, q\_min, q\_max, SWIGTYPE\_p\_unsigned\_long.getCPtr(q\_out));

}

public static SWIGTYPE\_p\_uint16\_t CMax\_uint16\_partition\_fuzzy(SWIGTYPE\_p\_uint16\_t vals, LongVector ids, long n, long q\_min, long q\_max, SWIGTYPE\_p\_unsigned\_long q\_out) {

return new SWIGTYPE\_p\_uint16\_t(swigfaissJNI.CMax\_uint16\_partition\_fuzzy\_\_SWIG\_0(SWIGTYPE\_p\_uint16\_t.getCPtr(vals), SWIGTYPE\_p\_long\_long.getCPtr(ids.data()), ids, n, q\_min, q\_max, SWIGTYPE\_p\_unsigned\_long.getCPtr(q\_out)), true);

}

public static SWIGTYPE\_p\_uint16\_t CMin\_uint16\_partition\_fuzzy(SWIGTYPE\_p\_uint16\_t vals, LongVector ids, long n, long q\_min, long q\_max, SWIGTYPE\_p\_unsigned\_long q\_out) {

return new SWIGTYPE\_p\_uint16\_t(swigfaissJNI.CMin\_uint16\_partition\_fuzzy\_\_SWIG\_0(SWIGTYPE\_p\_uint16\_t.getCPtr(vals), SWIGTYPE\_p\_long\_long.getCPtr(ids.data()), ids, n, q\_min, q\_max, SWIGTYPE\_p\_unsigned\_long.getCPtr(q\_out)), true);

}

public static SWIGTYPE\_p\_uint16\_t CMax\_uint16\_partition\_fuzzy(SWIGTYPE\_p\_uint16\_t vals, SWIGTYPE\_p\_int ids, long n, long q\_min, long q\_max, SWIGTYPE\_p\_unsigned\_long q\_out) {

return new SWIGTYPE\_p\_uint16\_t(swigfaissJNI.CMax\_uint16\_partition\_fuzzy\_\_SWIG\_1(SWIGTYPE\_p\_uint16\_t.getCPtr(vals), SWIGTYPE\_p\_int.getCPtr(ids), n, q\_min, q\_max, SWIGTYPE\_p\_unsigned\_long.getCPtr(q\_out)), true);

}

public static SWIGTYPE\_p\_uint16\_t CMin\_uint16\_partition\_fuzzy(SWIGTYPE\_p\_uint16\_t vals, SWIGTYPE\_p\_int ids, long n, long q\_min, long q\_max, SWIGTYPE\_p\_unsigned\_long q\_out) {

return new SWIGTYPE\_p\_uint16\_t(swigfaissJNI.CMin\_uint16\_partition\_fuzzy\_\_SWIG\_1(SWIGTYPE\_p\_uint16\_t.getCPtr(vals), SWIGTYPE\_p\_int.getCPtr(ids), n, q\_min, q\_max, SWIGTYPE\_p\_unsigned\_long.getCPtr(q\_out)), true);

}

public static void omp\_set\_num\_threads(int num\_threads) {

swigfaissJNI.omp\_set\_num\_threads(num\_threads);

}

public static int omp\_get\_max\_threads() {

return swigfaissJNI.omp\_get\_max\_threads();

}

public static SWIGTYPE\_p\_void memcpy(SWIGTYPE\_p\_void dest, SWIGTYPE\_p\_void src, long n) {

long cPtr = swigfaissJNI.memcpy(SWIGTYPE\_p\_void.getCPtr(dest), SWIGTYPE\_p\_void.getCPtr(src), n);

return (cPtr == 0) ? null : new SWIGTYPE\_p\_void(cPtr, false);

}

public static SWIGTYPE\_p\_float cast\_integer\_to\_float\_ptr(int x) {

long cPtr = swigfaissJNI.cast\_integer\_to\_float\_ptr(x);

return (cPtr == 0) ? null : new SWIGTYPE\_p\_float(cPtr, false);

}

public static SWIGTYPE\_p\_long cast\_integer\_to\_long\_ptr(int x) {

long cPtr = swigfaissJNI.cast\_integer\_to\_long\_ptr(x);

return (cPtr == 0) ? null : new SWIGTYPE\_p\_long(cPtr, false);

}

public static SWIGTYPE\_p\_int cast\_integer\_to\_int\_ptr(int x) {

long cPtr = swigfaissJNI.cast\_integer\_to\_int\_ptr(x);

return (cPtr == 0) ? null : new SWIGTYPE\_p\_int(cPtr, false);

}

public static void ignore\_SIGTTIN() {

swigfaissJNI.ignore\_SIGTTIN();

}

}